STIC Search Report
EIC 2100

STIC Database Tracking Number: 181657

TO:	From:
Thanh-ha Dang Location: RND 3B15 Art Unit: 2163 Thursday, March 09, 2006 Case Serial Number: 10/618409	Emory Damron Location: EIC 2100 RND 4B19 Phone: 571-272-3520 Emory.Damron@uspto.gov

Search Notes

Dear Thanh-ha,

Please find below your fast and focused results.
References of potential pertinence have been tagged, but please review all the packets in case you like something I didn't.
Of those references which have been tagged, please note any manual highlighting which I've done within the document.
In addition to searching on Dialog, I also searched EPO/JPO/Derwent, IEEE and Inspec.
There may be a few decent references contained herein, but I'll let you determine how useful they may be to you.
Please contact me if I can refocus or expand any aspect of this case, and please take a moment to provide any feedback (on the form provided) so EIC 2100 may better serve your needs. Good Luck!

Sincerely,
Emory Damron
Technical Information Specialist
EIC 2100, US Patent & Trademark Office
Phone: (571) 272-3520
Emory.damron@uspto.gov

```
? d s
Set     Items   Description
S1      34073   S (QUERY? OR QUERIE?)()LANGUAG? OR SQL? ? OR (XML? ? OR EXTEN?()(MARKUP?
OR MARK?()UP)()LANGUAG?)()QUER?
S2        289   S HQL? ? OR QUERYLANGUAG? OR GQL? ?
S3       9652   S (INTERROG? OR RETRIEV? OR OBTAIN? OR REQUIR? OR QUESTION? OR ENQUIR? OR
KEYWORD? OR KEY()WORD?)()LANGUAG? OR FETCH?
S4    6910026   S LINK? OR JOIN? OR COMBINE? OR COMBINING? OR COMBINATION? OR COMBINATOR?
OR RELATIONSHIP?
S5    7990551   S AFFILIAT? OR CONJOIN? OR RELATED OR RELATION? OR BUILD? OR CONCATENAT?
OR COMPIL?
S6    4314563   S KINSHIP? OR CORRESPOND? OR CORRELAT? OR DRAG?(2W)DROP? OR DRAGANDDROP?
OR DRAGNDROP?
S7   11210576   S ADAPT? OR RECONFIGUR? OR CONFIGUR? OR CHANGE? OR CHANGING? OR MODIF? OR
ALTER? OR CONVERT? OR CONVERSION?
S8    3119003   S AMEND? OR UPDAT? OR REVIS? OR REARRANG? OR REORDER? OR RESHUFFL? OR
RESET? OR REFORMAT? OR TRANSFORM?
S9   13781657   S FIELD? OR ARRAY? OR MENU? ? OR DROPDOWN? OR DROP()DOWN? OR GUI? ? OR
(GRAPHIC? OR USER?)(2W)INTERFACE? OR UI OR MODEL?
S10     46506   S INTERFACE?(2N)(APPLICATION? OR SOFTWARE?) OR DIALOG()(BOX OR BOXES) OR
PULLDOWN? OR PULL?()DOWN? OR QAUI? ?
S11     91404   S API? ? OR (APP OR APPS OR APPLICATION?)(2N)(INTERFACE? OR SCREEN? OR
WINDOW? OR PAGE? ? OR BOX OR CELL? ? OR TEMPLAT?)
S12     26980   S S1:S3 AND S4:S8
S13     17783   S S1:S3 AND S9:S11
S14     32526   S S12:S13
S15     12146   S USER? OR CLIENT? OR CUSTOMER? OR ENDUSER? OR NETIZEN? OR OPERATOR? OR
MANAGER?
S16       466   S ACCOUNT?(2N)HOLDER? OR PATRON? OR MEMBER? OR SUBSCRIBER? OR WEBUSER?
S17      1487   S PARTY? OR PERSON? ? OR INDIVIDUAL? OR PARTIE? OR PRINCIPAL?
S18      5276   S DESIRED? OR CHOICE? OR PREFER? OR CHOOS? OR SELECT? OR ELECT? OR OPT? ?
OR OPTING OR OPTED OR OPTION? OR PRECEDEN?
S19       696   S CUSTOMIZ? OR CUSTOMIS? OR PERSONALIS? OR PERSONALIZ? OR INDIVIDUALIZ? OR
INDIVIDUALIS?
S20      2505   S ELECT? OR PICK? OR DESIGNAT? OR DISCRIMINAT? OR ASSIGN? OR SPECIFY? OR
PRESET?
S21       769   S STATEMENT?
S22     10153   S STRING? OR SET? ? OR SEQUENC? OR BYTE? OR BIT? ? OR CHARACTERS OR LINE?
? OR STREAM? OR PREDICATE?
S23      5175   S CHUNK? ? OR TEXT? OR SEGMENT? OR SENTENC? OR SCRIPT? OR EXPRESSION? OR
QUEUE? OR CLAUSE? OR DECLARATION?
S24     13964   S LOGICAL? OR ABSTRACT? OR OBJECT? OR DATAOBJECT? OR METADATA? OR
META()DATA? OR VIRTUAL? OR HTML(2N)DATA? OR HTMLOBJECT? OR CONCEPTUAL?
S25      4555   S PHYSICAL? OR REAL? OR EXECUTAB? OR TANGIB? OR RUNNABLE? OR (RUN? ? OR
LAUNCH? OR START?)(2N)CAPAB? OR LAUNCHAB?
S26       313   S USABL? OR UTILIZAB? OR UTILISAB? OR UTILE OR CONCRETE?
S27     12237   S S14 AND S12 AND S13
S28      1061   S S14 AND S15:S17 AND S18:S20 AND S21:S23 AND S24:S26
S29       529   S S27 AND S28
S30        92   S S29 AND S24 AND S25:S26
S31       147   S S28 AND S24 AND S25:S26
S32       147   S S30:S31
S33       120   S S32 AND PY<2004
S34       122   S S32 NOT PY>2003
S35       122   S S33:S34
S36       117   RD  (unique items)
 ; show files
```

[File 2] **INSPEC** 1898-2006/Feb W4

36/3,K/89 (Item 11 from file: 35) **Links**
Dissertation Abs Online

01458959 ORDER NO: AADAA-I1375595

ENHANCED QUERY FORMULATION AND DATABASE MANAGEMENT TOOL DESIGN

Author: BADHE, HARISH
Degree: M.S.C.S.
Year: 1994
Corporate Source/Institution: UNIVERSITY OF LOUISVILLE (0110)
Source: Volume 34/01 of MASTERS ABSTRACTS. of Dissertations Abstracts International.
PAGE 337 .
Year: 1994

...uses. A database requires the ability to manage the various entities associated with it (viz. **users**, accounts, **physical** and **logical** space, privileges, structure etc.). All these actions are effected by **user**-actions generically termed as query **statements**. The act of querying assumes particular importance in **relational** databases which advocate a universal **relationship** concept, since the **user** has to **build** the **relations** each time he/she queries the database. The complexity of each query depends on the underlying data structure and the number of **relations** the **user** wants to **specify**. This thesis developes a **graphic interface** tool which replaces the bare command-**line** interface provided by most database servers can aid in better query formulation and focuses the **user**'s attention on problems other than remembering the syntax of fourth generation **query languages** like Structured **Query Languages(SQL)**. The test-bed chosen for developing this interface is ORACLE 6.0. Also keeping in touch with the latest trend in graphics development, Visual Basic 3.0 was used to **build** the **user-interface**. The front-end query-tool is designed to accommodate **users** at extreme ends of the learning curve as well as everyone in between.

36/3,K/92 (Item 14 from file: 35) Links
Dissertation Abs Online

01375367 ORDER NO: AAD94-26045

A FORMAL DEFINITION, QUERY LANGUAGE AND RELATIONAL DATABASE MAPPING FOR THE SEMANTIC DATABASE MODEL (SDM)

Author: AL-MADANI, NABEIL ISMAEIL
Degree: PH.D.
Year: 1994
Corporate Source/Institution: UNIVERSITY OF SOUTH FLORIDA (0206)
Source: Volume 5505B of Dissertations Abstracts International.
PAGE 1910 . 195 PAGES
A FORMAL DEFINITION, QUERY LANGUAGE AND RELATIONAL DATABASE MAPPING FOR THE SEMANTIC DATABASE MODEL (SDM)
Year: 1994

Conventional database models, particularly the relational model, provided data independence by separating the conceptual representation of data from the physical implementation. However, these models lack the constructs and mechanisms for modeling the complex interrelationships among data items in a real-world application environment. Semantic database models have been developed to provide a richer set of data structuring constructs and mechanisms. The Semantic Database Model (SDM) is one of the most comprehensive published semantic database models. It incorporates a wide range of constructs for expressing data abstractions, relationships among data items, and data derivation mechanisms. The SDM, however, is defined by means of examples and does not have a formally defined semantics or query language. The primary focus of this dissertation is: (1) to formally define the SDM, (2) to develop a query language for the SDM, (3) to develop a scheme that maps the structures of the SDM to relational structures, and (4) to design and develop a prototype system that implements the SDM as a front-end to a relational DBMS. The formal definition of the SDM uses a graph-based formalism for defining the structural components, and first order predicate calculus for specifying the integrity constraints of the model. The SDM query language is an English-like query language that provides commands for schema definition and modification, data retrieval and manipulation, and data update and modification. The mapping scheme provides the necessary theoretical foundations for developing the SDM Schema Management System (SDM/SMS) prototype. The SDM/SMS implements the SDM on top of a relational DBMS. It allows the user to define and manipulate schemata, browse and navigate inside schemata, and specify queries through an intuitive yet powerful graphical user interface.

36/3,K/66 (Item 41 from file: 8) **Links**
Fulltext available through: USPTO Full Text Retrieval Options
Ei Compendex(R)

03755904 **E.I. No:** EIP93121142488
Title: Effective utilization of copies in a transparent distributed environment

Author: Orlowska, Maria E.
Corporate Source: Univ of Queensland, St Lucia, Aust
Source: Distributed and Parallel Databases v 1 n 4 Oct 1993. p 409-425
Publication Year: 1993
CODEN: DPADEH **ISSN:** 0926-8782
Language: English
Abstract: In a distributed **relational** database system, the processing of a query involves data transmission among different sites via a computer network. In a distributed database multiple copies of each **relation** can be allocated to different, **physically** distributed sites. In this paper we discuss the query preoptimization problem for **join**-queries. In general, there is a large number of possibilities to use the copies of the data item in a distributed **relational** database when evaluating a **join**-query. We consider the problem of a copy preselection for each **relation** in a **join sequence** of a **join**-query. We show how to express the preselection problem for a given query and data... ...cover problem. It can be treated as a heuristic for the first phase of a **join**-query optimization, and as such as an input to the final stage of optimization, the execution strategy generation for a **join**-query. In this paper we assumed that a distributed system provides fully transparent data management, i.e., data allocation to the network and data replication which is revealed to a **user**. We illustrate the proposed mathematical programming problem through a nontrivial example. (Author **abstract**) 18 Refs.
Descriptors: *Distributed database systems; **Query languages**; Data communication systems; **Relational** database systems; Computer networks; Linear programming; Heuristic methods; **User interfaces**; Computer **software selection** and evaluation ; Mathematical programming
Identifiers: Copy preselection; Transparent distributed environment; Query optimization; **Join** query; Data allocation; Integer linear programming; Replicated data

36/3,K/72 (Item 2 from file: 34) **Links**
Fulltext available through: USPTO Full Text Retrieval Options
SciSearch(R) Cited Ref Sci

04320045 **Genuine Article#:** RV576 **No. References:** 20
AN EXECUTABLE VISUAL FORMALISM FOR OBJECT-ORIENTED CONCEPTUAL MODELING

Author: KUNG DC
Corporate Source: UNIV TEXAS,DEPT COMP SCI ENGN,POB 19015/ARLINGTON//TX/76019
Journal: JOURNAL OF SYSTEMS AND SOFTWARE , **1995** , V 31 , N1 (OCT) , P 33-43
ISSN: 0164-1212
Language: ENGLISH **Document Type:** ARTICLE (Abstract Available)
AN EXECUTABLE VISUAL FORMALISM FOR OBJECT-ORIENTED CONCEPTUAL MODELING , 1995

Abstract: Conceptual modeling aims at establishing the conceptual knowledge necessary for proper communication between a development team and users. This article presents an executable visual formalism for object-oriented modeling of information systems. This formalism is an integration of the entity-relationship approach, Petri nets, relational calculus, and time temporal logic. It supports integrated and encapsulated modeling of the structural and behavioral aspects of objects, and object evolution. The formalism has textual and graphical representations, allows formal analysis of model properties, and supports rapid prototyping. An environment and a methodology for conceptual modeling also are described.
Research Fronts: 93-0083 001 (DISTRIBUTED DATABASE DESIGN; OBJECT MODEL; EXTENDED NESTED RELATIONS)
93-1567 001 (OBJECT -ORIENTED SOFTWARE SYSTEMS; DESIGN EVOLUTION; BEHAVIOR NETWORK MODEL FOR CONCEPTUAL INFORMATION MODELING)
93-1976 001 (OBJECT-ORIENTED MODELING; DICTIONARY OF ENTITY- RELATIONSHIP DATA SCHEMAS; SOFTWARE ENGINEERING ENVIRONMENT)
93-4292 001 (FUNDAMENTAL GRAPHICAL PRIMITIVES FOR VISUAL QUERY LANGUAGES; SPECIFYING MULTIPLE-VIEWED SOFTWARE REQUIREMENTS)
93-5836 001 (REACTIVE SYSTEMS; REAL-TIME SPECIFICATION USING PETRI NETS; ABSTRACT DESIGN REPRESENTATIONS; COMPOSITIONAL SEMANTICS; DISTRIBUTED PROTOTYPING)

36/3,K/8 (Item 8 from file: 2) **Links**
INSPEC

07325423 **INSPEC Abstract Number:** C1999-09-6160M-021
Title: Specifying generic multimedia 3D visualizations and temporal presentations from database queries
Author Baral, C.; Gonzalez, G.
Author Affiliation: Dept. of Comput. Sci., Texas Univ., El Paso, TX, USA
Conference Title: Proceedings IEEE International Conference on Multimedia Computing and Systems **Part** vol.1 p. 550-5 vol.1
Publisher: IEEE Comput. Soc , Los Alamitos, CA, USA
Publication Date: 1999 **Country of Publication:** USA 2 vol. (xlix+909+1127) pp.
ISBN: 0 7695 0253 9 **Material Identity Number:** XX-1999-01834
U.S. Copyright Clearance Center Code: 0 7695 0253 9/99/$10.00
Conference Title: Proceedings of ICMCS99: IEEE Multimedia Systems '99: International Conference on Multimedia Computing and Systems
Conference Sponsor: IEEE Comput. Soc.; IEEE Circuit & Syst. Soc.; IEEE Commun. Soc.; IEEE Signal Process. Soc
Conference Date: 7-11 June 1999 **Conference Location:** Florence, Italy
Language: English
Subfile: C

Title: Specifying generic multimedia 3D visualizations and temporal presentations from database queries
Abstract: The extended version of SQL+D presented here takes multimedia to its ultimate realm: the third dimension, allowing the user to explore the data in a virtual reality environment and display VRML files as data objects. Unlike the limited number of efforts in this area, SQL+D goes beyond constructing 3D bar charts. It allows truly interactive environments dynamically specified by the user in a query and constructed from data extracted from the database. Multimedia presentations are usually specified for a fixed set of elements. When dealing with database queries, the specific elements are unknown until the answer set is obtained. We present the syntax of an improved version of SQL+D that allows a user to specify complex generic temporal presentations within the query itself. We have implemented most of our proposed...
Descriptors: ...SQL; ... ...user interfaces; ... ...virtual reality languages
Identifiers: ...virtual reality environment... ...data objects; ... ...SQL+D... ...answer set;
1999

36/3,K/13 (Item 13 from file: 2) **Links**
INSPEC

06056522 **INSPEC Abstract Number:** C9511-6160J-011
Title: METU object-oriented DBMS
Author Dogac, A.; Ozkan, C.; Arpinar, B.; Okay, T.; Evrendilek, C.
Author Affiliation: Centre of Software Res. & Dev., Middle East Tech. Univ., Ankara, Turkey
Conference Title: Advances in Object-Oriented Database Systems. Proceedings of the NATO Advanced Study Institute p. 327-54
Editor(s): Dogac, A.; Ozsu, M.T.; Biliris, A.; Sellis, T.
Publisher: Springer-Verlag , Berlin, Germany
Publication Date: 1994 **Country of Publication:** West Germany xi+515 pp.
ISBN: 3 540 57825 0
Conference Title: Proceedings of NATO Advanced Study Institute on Object- Oriented Databases
Conference Sponsor: NATO
Conference Date: 6-16 Aug. 1993 **Conference Location:** Kusadasi, Turkey
Language: English
Subfile: C

Title: METU object-oriented DBMS
Abstract: METU object-oriented DBMS includes the implementation of a database kernel, an object-oriented SQL-like language and a graphical user interface. Kernel functions are divided between a SQL interpreter and a C++ compiler. Thus the interpretation of functions are avoided increasing the efficiency of the system. The compiled by C++ functions are used by the system through the function manager. The system is realized on Exodus Storage Manager (ESM), thus exploiting some of the kernel functions readily provided by ESM. The additional functions provided by the MOOD kernel are the optimization and interpretation of SQL statements, dynamic linking of functions, and catalog management. An original query optimization strategy based on the object-oriented features of the language is developed. For this purpose formulas for the selectivity of a path expression, and for the cost of forward and backward path traversals are derived, and join sizes are estimated. New strategies for ordering the joins and path expressions are also developed. A graphical user interface, namely MoodView is implemented on the MOOD kernel. MoodView provides the database programmer with tools... ...for every phase of OODBMS application development. Furthermore, a database administration tool, a full screen text-editor, a SQL based query manager, and a graphical indexing tool for the spatial data, i.e., R Trees are also...
Descriptors: graphical user interfaces; ... ...object-oriented databases... ...object-oriented languages... ...program compilers; ... ...query languages; ... ...SQL
Identifiers: ...object-oriented database... ...object-oriented query language; ... ...graphical user interface; ... ...SQL interpreter... ...C++ compiler; ... ...Exodus Storage Manager; ... ...full screen text-editor
1994

```
? d s
Set     Items   Description
S1       1555   S (QUERY? OR QUERIE?)()LANGUAG? OR SQL? ? OR (XML? ? OR EXTEN?()(MARKUP?
OR MARK?()UP)()LANGUAG?)()QUER?
S2          7   S HQL?. ? OR QUERYLANGUAG?
S3        192   S (INTERROG? OR RETRIEV? OR OBTAIN? OR REQUIR? OR QUESTION? OR ENQUIR? OR
KEYWORD? OR KEY()WORD?)()LANGUAG?
S4    2130087   S LINK? OR JOIN? OR COMBINE? OR COMBINING? OR COMBINATION? OR COMBINATOR?
OR RELATIONSHIP?
S5    1168008   S AFFILIAT? OR CONJOIN? OR RELATED OR RELATION? OR BUILD?
S6    1472627   S KINSHIP? OR CORRESPOND? OR CORRELAT? OR DRAG?(2W)DROP? OR DRAGANDDROP?
OR DRAGNDROP?
S7    3787360   S ADAPT? OR RECONFIGUR? OR CONFIGUR? OR CHANGE? OR CHANGING? OR MODIF? OR
ALTER? OR CONVERT? OR CONVERSION?
S8     628811   S AMEND? OR UPDAT? OR REVIS? OR REARRANG? OR REORDER? OR RESHUFFL? OR
RESET? OR REFORMAT? OR TRANSFORM?
S9       1292   S S1:S3 AND S4:S8
S10   1028343   S FIELD? OR ARRAY? OR MENU? ? OR DROPDOWN? OR DROP()DOWN? OR GUI? ? OR
(GRAPHIC? OR USER?)(2W)INTERFACE? OR UI OR MODEL?
S11     17241   S INTERFACE?(2N)(APPLICATION? OR SOFTWARE?) OR DIALOG()(BOX OR BOXES) OR
PULLDOWN? OR PULL?()DOWN?
S12     11228   S API? ? OR (APP OR APPS OR APPLICATION?)(2N)(INTERFACE? OR SCREEN? OR
WINDOW? OR PAGE? ? OR BOX OR CELL? ?)
S13       371   S S1:S3 AND S10:S12
S14      1379   S S9 OR S13
S15       701   S USER? OR CLIENT? OR CUSTOMER? OR ENDUSER? OR NETIZEN? OR OPERATOR? OR
MANAGER?
S16        41   S ACCOUNT?(2N)HOLDER? OR PATRON? OR MEMBER? OR SUBSCRIBER? OR WEBUSER?
S17        86   S PARTY? OR PERSON? ? OR INDIVIDUAL? OR PARTIE? OR PRINCIPAL?
S18       379   S DESIRED? OR CHOICE? OR PREFER? OR CHOOS? OR SELECT? OR ELECT? OR OPT? ?
OR OPTING OR OPTED OR OPTION? OR PRECEDEN?
S19        24   S CUSTOMIZ? OR CUSTOMIS? OR PERSONALIS? OR PERSONALIZ? OR INDIVIDUALIZ? OR
INDIVIDUALIS?
S20       178   S ELECT? OR PICK? OR DESIGNAT? OR DISCRIMINAT? OR ASSIGN? OR SPECIFY? OR
PRESET?
S21       266   S STATEMENT?
S22       467   S STRING? OR SET? ? OR SEQUENC? OR BYTE? OR BIT? ? OR CHARACTERS OR LINE?
? OR STREAM? OR PREDICATE?
S23       441   S CHUNK? ? OR TEXT? OR SEGMENT? OR SENTENC? OR SCRIPT? OR EXPRESSION? OR
QUEUE? OR CLAUSE?
S24       367   S LOGICAL? OR ABSTRACT? OR OBJECT? OR DATAOBJECT? OR METADATA? OR
META()DATA?
S25       138   S PHYSICAL? OR REAL? OR EXECUTAB? OR TANGIB? OR RUNNABLE? OR (RUN? ? OR
LAUNCH? OR START?)(2N)CAPAB? OR LAUNCHAB?
S26         8   S USABL? OR UTILIZAB? OR UTILISAB? OR UTILE
S27      1298   S IC=G06F?
S28      1061   S MC=T01?
S29       284   S S9 AND S13
S30       284   S S29 AND S15:S28
S31       316   S S14 AND S15:S17 AND S18:S20
S32        51   S S14 AND (S21:S23 OR S10:S12) AND S24 AND S25:S26
S33        80   S S30 AND S31
S34       265   S S30:S31 AND S21:S23 AND S1:S3 AND S4:S6
S35       259   S S34 AND S27:S28
S36       165   S (S34:S35 OR S14) AND S24:S26 AND S15:S17 AND S21:S23
S37        82   S S36 AND S34:S35
S38       175   S S32:S33 OR S37
S39       110   S S38 AND AC=US/PR
S40        95   S S39 AND AY=(1970:2003)/PR
S41        89   S S39 NOT AY=(2004:2006)/PR
```

```
S42          65    S S38 NOT S39
S43          58    S S42 AND PY=1970:2003
S44          53    S S42 NOT PY=2004:2006
S45         156    S S40:S41 OR S43:S44
S46         156    IDPAT (sorted in duplicate/non-duplicate order)
 ; show files
```

[File 347] **JAPIO** Nov 1976-2005/Nov(Updated 060302)

[File 350] **Derwent WPIX** 1963-2006/UD,UM &UP=200616

**File 350: For more current information, include File 331 in your search. Enter HELP NEWS 331 for details.*

46/3,K/129 (Item 129 from file: 350) **Links**
Derwent WPIX

008823290 **Image available**
WPI Acc No: 1991-327303/199145
XRPX Acc No: N91-250706

Object-oriented query language appts. - .includes pre-processor to parse, optimise and translate oql statements into host language statements which are compiled into executable code

Patent Assignee: TEXAS INSTR INC (TEXI)
Inventor: BLAKELY J A; THOMPSON C W; BLAKELEY J A
Number of Countries: 006 Number of Patents: 006
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 455447	A	19911106	EP 91303851	A	19910429	199145	B
EP 455447	A3	19930630	EP 91303851	A	19910429	199405	
US 5761493	A	19980602	US 90516369	A	19900430	199829	
US 5826077	A	19981020	US 90516369	A	19900430	199849	
			US 95473622	A	19950607		
			US 96639808	A	19960429		
			US 97837396	A	19970417		
EP 455447	B1	19990616	EP 91303851	A	19910429	199928	
DE 69131336	E	19990722	DE 631336	A	19910429	199935	
			EP 91303851	A	19910429		

Priority Applications (No Type Date): US 90516369 A 19900430; US 95473622 A 19950607; US 96639808 A 19960429; US 97837396 A 19970417
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 455447	A		35		
Designated States (Regional): DE FR GB IT NL					
EP 455447	A3		35		
US 5826077	A			G06F-017/30	Cont of application US 90516369
					Cont of application US 95473622
					Cont of application US 96639808
EP 455447	B1	E		G06F-017/60	
Designated States (Regional): DE FR GB IT NL					
DE 69131336	E			G06F-017/60	Based on patent EP 455447
US 5761493	A			G06F-017/30	

Object-oriented query language appts...

...includes pre-processor to parse, optimise and translate oql statements into host language statements which are compiled into executable code

...Abstract (Basic): The appts. for processing queries includes a preprocessor (144, 148) to parse, optimise and translate **object query language statements** into efficient lost

language **statements**, connected to a compiler (152) that compiles such host language **statements** into **executable** code (154). A programme is inputted containing **statements** using a host language's type system as well as employing the host language's **expressions**, **object** composition and inheritance in the formulation of queries...

...The Select-From-Where style of standard **query language** is **combined** as a basic structure for query **statements** with the **set**-valued function of the host language as targets for those queries. Thus, an associative query...

Title Terms: **OBJECT**;

-continued

```
| function_expression
| parameter_commalist ',' object_ref
;
```

The approach used in OQL to provide a better integration with host languages can also be used by current SQL systems to provide a better integration with object-oriented programming languages.

The invention provides an elegant way to combine the computer industry's most used industrial strength object-oriented programming language, C++, with the industry's standard database query language SQL. The invention complements a C++ programmer's existing programming tools, by adding a query capability. This can result in better, more reliable C++ code and make database programming more accessible to the large number of C++ programmers.

The query processor architecture of OQL **100** is illustrated in FIG. **7**. It consists of four main modules: parser **102**, query rewriter **104**, rule-based query translator and optimizer **106**, and code optimizer **108**. All these modules use the run-time data dictionary **112** that contains all type information of the application including class information (class names, data member names and types, and member function names types and parameters), class hierarchy information, and set-valued and Boolean-valued variables and functions used in the applications. The query processor takes as input OQL[C++]'s SELECT statements **124**, and produces as output optimized C++ or Persistent C++ code **126**. In FIG. **7**, lightly shadowed boxes represent run-time data structures, darker boxes represent rule-bases, arrows connecting modules represent transfer of control, and continuous lines represent access to data structures from the corresponding modules. Parser **102** takes as input OQL[C++] SELECT statement and, with the help of the data dictionary, checks the syntax of the statement and whether references to class information and user-defined functions are valid. Syntactically incorrect statements are issued syntax errors **122** and syntactically correct statements are mapped into a query graph **116**. After parsing, control passes to query rewrite module **104** which reads query graph **116** and transforms it in place after performing semantic and algebraic query optimization with the help of data dictionary **112** and algebraic rule-base **110**, respectively. After query rewrite, control passes to rule-based query translator **106** which with the help of storage interface rule-base **118** and data dictionary **112** translates the query graph into a first version of an execution plan **114** written in C++ or Persistent C++. Finally, code optimizer **108** produces an improved sequence of C++ and Persistent C++ code **126** statements corresponding to original query **124**.

The complete compilation process **140** of a program containing OQL[C++] statements (i.e., DECLARE, IMPLEMENT, and SELECT statements) is illustrated in FIG. **8**. The input to the compilation process is a program containing OQL[C++] statements **142**. This input is processed by the OQL preprocessor which, for SELECT statements, performs all the functions described in query processor **100** (FIG. **7**). The OQL preprocessor produces a combination of C++ and Persistent C++ code **146** because of the possibility of a program querying transient and persistent data. A persistence C++ preprocessor **148** transforms Persistent C++ code into pure C++ code **150** and C++ code passes this process unchanged. C++ code **150** is then processed by a standard C++ compiler **152** which produces executable code **154**.

FIG. **9** illustrates a possible placement of an application code **164** containing OQL[C++] query statements **166**, according to a preferred embodiment of the invention, in the main memory **162** of a computer **160**. During the development of the application, a computer prograrmmer enters a program to the computer **160** via a computer terminal **180**. The application is compiled according to the process described in FIG. **8** where OQL[C++] statements **142** (FIG. **8**)/**166** (FIG. **9**) are translated into pure C++ code **168** or Persistent C++ code **178** depending on whether queries access transient **170** or persistent data **172**. Persistent data **172** is stored in a persistent database **176** on disk **174**. Queries on transient data are performed using C++ code **168** in main memory **170**. Queries on persistent data are performed using Persistent C++ code **178** which ensures the transfer of persistent data from the database **176** on disk **174** to a special area **172** of main memory **162** where the query is executed. It should be noted that although application code **164**, OQL[C++] code **166**, C++ code **168**, and Persistent C++ code **178** are distinguishable before compilation, they are a single, indistinguishable piece of execution code after compilation.

The description of OQL[C++] as implemented by a a C++ preprocessor is specific to the implemented embodiment. If native C++ compilers (not implemented by preprocessors) were extended to use OQL, similar operations would be implemented by those compilers. In fact, if incremental C++ compilers become available, then OQL[C++] will be able to be similarly incremental and still inherit the "ad hoc" interactive ability to state queries. This feature would be inherited immediately in OQL[CLOS] when applied according to the present invention since CLOS allows incremental compilation.

While a specific embodiment of the invention has been shown and described above in this patent application, various modifications and alternate embodiments will occur to those skilled in the art. Accordingly, it is intended that the invention be limited only in terms of the appended claims.

We claim:

1. A method for enabling an integration of an object query language for an object-oriented data model with a host language, wherein queries for accessing objects in the object oriented data model are in said object query language, comprising the steps of:

constructing a preprocessor including a grammar coupling the object query language with the host language, wherein said grammar includes query statements to use expressions of the host language in accessing the objects and wherein said query statements access both transient and persistent objects by accessing a data dictionary having pointers to said transient and persistent objects to maintain information corresponding to both said transient and persistent objects;

using said preprocessor to preprocess said query statements by parsing, optimizing and translating at least one statement in said object query language into at least one statement of the host language; and

compiling said statement of the host language into executable code.

2. A method for executing a query in an object query language operating in conjunction with a host language, comprising the steps of:

defining the object query language with set declarations;

formulating query statements by combining a type system of the host language as a database model with preselected aspects of a relational query language as a basic structure for said query statements;

preprocessing said query statements into host language statements, said host language statements to access

and Boolean-valued variables and functions used in the applications. The query processor takes as input OQL[C++]'s SELECT statements **124**, and produces as output optimized C++ or Persistent C++ code **126**. In FIG. **4**, lightly shadowed boxes represent run-time data structures, darker boxes represent rule-bases, arrows connecting modules represent transfer of control, and continuous lines represent access to data structures from the corresponding modules. Parser **102** takes as input OQL[C++] SELECT statement and, with the help of the data dictionary, checks the syntax of the statement and whether references to class information and user-defined functions are valid. Syntactically incorrect statements are issued syntax errors **122** and syntactically correct statements are mapped into a query graph **116**. After parsing, control passes to query rewrite module **104** which reads query graph **116** and transforms it in place after performing semantic and algebraic query optimization with the help of data dictionary **112** and algebraic rule-base **110**, respectively. After query rewrite, control passes to rule-based query translator **106** which with the help of storage interface rule-base **118** and data dictionary **112** translates the query graph into a first version of an execution plan **114** written in C++ or Persistent C++. Finally, code optimizer **108** produces an improved sequence of C++ and Persistent C++ code **126** statements corresponding to original query **124**.

The complete compilation process **140** of a program containing OQL[C++] statements (i.e., DECLARE, IMPLEMENT, and SELECT statements) is illustrated in FIG. **5**. The input to the compilation process is a program containing OQL[C++] statements **142**. This input is processed by the OQL preprocessor which, for SELECT statements, performs all the functions described in query processor **100** (FIG. **4**). The OQL preprocessor produces a combination of C++ and Persistent C++ code **146** because of the possibility of a, program querying transient and persistent data. A persistence C++ preprocessor **148** transforms Persistent C++ code into pure C++ code **150** and C++ code passes this process unchanged. C++ code **150** is then processed by a standard C++ compiler **152** which produces executable code **154**.

FIG. **6** illustrates a possible placement of an application code **164** containing OQL[C++] query statements **166**, according to a preferred embodiment of the invention, in the main memory **162** of a computer **160**. During the development of the application, a computer programmer enters a program to the computer **160** via a computer terminal **180**. The application is compiled according to the process described in FIG. **5** where OQL[C++] statements **142** (FIG. **5**)/**166** (FIG. **6**) are translated into pure C++ code **168** or Persistent C++ code **178** depending on whether queries access transient **170** or persistent data **172**. Persistent data **172** is stored in a persistent database **176** on disk **174**. Queries on transient data are performed using C++ code **168** in main memory **170**. Queries on persistent data are performed using Persistent C++ code **178** which ensures the transfer of persistent data from the database **176** on disk **174** to a special area **172** of main memory **162** where the query is executed. It should be noted that although application code **164**, OQL[C++] code **166**, C++ code **168**, and Persistent C++ code **178** are distinguishable before compilation, they are a single, indistinguishable piece of execution code after compilation.

The description of OQL[C++] as implemented by a a C++ preprocessor is specific to the implemented embodiment. If native C++ compilers (not implemented by preprocessors) were extended to use OQL, similar operations would be implemented by those compilers. In fact, if incremental C++ compilers become available, then OQL[C++] will be able to be similarly incremental and still inherit the "ad hoc" interactive ability to state queries. This feature would be inherited immediately in OQL[CLOS] when applied according to the present invention since CLOS allows incremental compilation.

While a specific embodiment of the invention has been shown and described above in this patent application, various modifications and alternate embodiments will occur to those skilled in the art. Accordingly, it is intended that the invention be limited only in terms of the appended claims.

We claim:

1. A method enabling an integration of an object query language for an object-oriented data model with a host language, wherein queries for accessing objects are in said object query language, comprising the steps of:

constructing a preprocessor including a grammar coupling the object query language with the host language, wherein said grammar includes query statements to use expressions of the host language in accessing the objects and wherein said query statements access both transient and persistent objects;

using said preprocessor to preprocess said query statements by parsing, optimizing and translating at least one statement in said object query language into at least one statement of the host language; and

compiling said statement of the host language into executable code.

2. The method of claim **1** wherein said preprocessing step includes the step of global type checking of said query statements.

3. A method for executing a query in an object query language operating in conjunction with a host language, comprising the steps of:

defining the object query language with set declarations;

formulating query statements by combining a type system of the host language as a database model with preselected aspects of a relational query language as a basic structure for said query statements;

preprocessing said query statements into host language statements, said host language statements being adapted to access transient and persistent objects;

compiling said host language statements; and

executing said compiled host language statements to return at least one result in response to said formulated query statement.

4. The method of claim **3**, wherein the method further comprises the step of defining said set declarations to include at least one membership function.

5. The method of claim **3**, wherein the method further comprises the step of defining said set declarations to include at least one set operation.

6. The method of claim **3**, wherein the method further comprises the step of defining said set declarations to include at least one iteration operator.

7. The method of claim **3**, wherein the method further comprises the step of defining said set declarations to include at least one indexing function.

8. The method of claim **3**, wherein said step of formulating query statements further comprises the step of combining boolean-valued functions of the host language with at least one predicate.

9. The method of claim **3**, wherein said step of formulating query statements farther comprises the step of setting targets for said queries by using set-valued functions of the host language.

46/3,K/114 (Item 114 from file: 350) **Links**
Derwent WPIX

011109053
WPI Acc No: 1997-086978/199708
XRPX Acc No: N97-071730
GUI for database retrieval - generates retrieval instructions of query language program by set creating, drawing, selection, projection, and extraction, and graphs quantity of and logical relation between sets retrieved
Patent Assignee: NAGAMORI N (NAGA-I)
Inventor: NAGAMORI N
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 5592663	A	19970107	US 93168219	A	19931217	199708	B

Priority Applications (No Type Date): US 93168219 A 19931217
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5592663	A			G06F-017/30	

GUI for database retrieval...

...**generates retrieval instructions of query language program by set creating, drawing, selection, projection, and extraction, and graphs quantity of and logical relation between sets retrieved**

...Abstract (Basic): The program communicates with the database **query language** program through message exchanges between the programs or message exchanges at the terminal systems connected with the communication network. The retrieval instructions of the **query language** program are generated by way of **set** creating, **set** drawing, **set selection, set** projection, and **set** extraction. The input operation is performed in place of the **user.**

...

...ADVANTAGE - **Graphical user interface** shows with graphs, quantity of and **logical relation** between **sets** of database retrieved, to calculate and manipulate **sets** by pointing area of rectangular shapes of graphs, and to extract data by pointing area of rectangular shapes of graphs. Facilitates simple retrieval processing and supports **logical** thinking of **user** to evaluate and judge quantity of **set** and **relation** between **sets.**

...Title Terms: **SET**;
International Patent Class (Main): **G06F-017/30**
Manual Codes (EPI/S-X): **T01-J05B3**

US005592663A

United States Patent [19]

Nagamori

[11] Patent Number: 5,592,663

[45] Date of Patent: Jan. 7, 1997

[54] **GRAPHING METHOD AND APPARATUS FOR DATA BASE RETRIEVAL**

[76] Inventor: **Nobuhiko Nagamori**, 5-5-13, Kunugidaidanchi, 1404, Kawashima-cho, Hodogaya-ku, Yokohama-shi, Kanagawa 240, Japan

[21] Appl. No.: **168,219**

[22] Filed: **Dec. 17, 1993**

[51] **Int. Cl.**[6] **G06F 17/30**

[52] **U.S. Cl.** **395/605**; 364/488; 364/283.4; 364/DIG. 1

[58] **Field of Search** 395/600, 100, 395/575; 364/468, 401, 488

[56] **References Cited**

U.S. PATENT DOCUMENTS

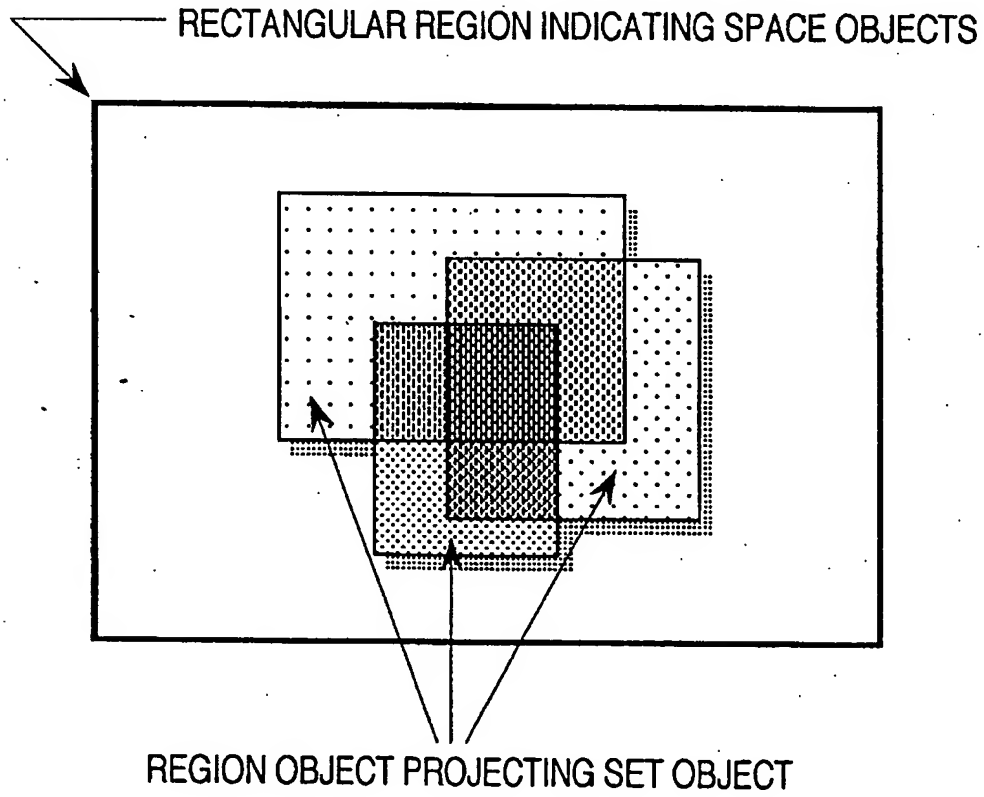
4,769,636	9/1988	Iwami et al.	395/158
4,829,446	5/1989	Draney	364/488
4,843,569	6/1989	Sawada et al.	382/24
4,845,634	7/1989	Vitek et al.	364/468
4,847,788	7/1989	Shimada et al.	395/135
4,882,679	11/1989	Tuy et al.	364/413.22
5,053,956	10/1991	Donald et al.	364/401
5,184,306	2/1993	Erdman et al.	395/119
5,278,946	1/1994	Shimada et al.	395/62
5,412,774	5/1995	Agrawal et al.	395/600
5,414,809	5/1995	Hogan et al.	395/155
5,421,008	5/1995	Banning et al.	395/600
5,448,696	9/1995	Shimada et al.	395/161
5,455,945	10/1995	VanderDrift	395/600

Primary Examiner—Thomas G. Black
Assistant Examiner—C. Pham
Attorney, Agent, or Firm—Ronald R. Snider

[57] **ABSTRACT**

The purpose of the present invention is to provide a graphical user interface which enables to show with graphs, as in FIG. 1, the quantity of and the logical relation between the sets of the database retrieved, to calculate and manipulate the sets by pointing the area of the rectangular shapes of the graphs, and to extract the data by pointing the area of the rectangular shapes of the graphs. The program of the present invention communicates with the database query language program through message exchanges between the programs or message exchanges at the terminal systems connected with the communication network, generates the retrieval instructions of the query language program by way of set creating, set drawing, set selection, set projection, and set extraction, and performs the input operation in place of the user. The present invention facilitates simple retrieval processing and supports logical thinking of the user to evaluate and judge the quantity of the set and the relation between the sets.

20 Claims, 15 Drawing Sheets



As shown in FIG. 14, the read set button (readSet) changes to "OK?" and a message "Select a group of fields to extract from the field object browser" is shown in the notifying field. Click the fields to be extracted, from the field object browser.

```
072 001-9I:read Items=:personal_id, name, age, dept_no,
073 001-1C:readSet <+>(('A'*'B'*'D')+('A'*'C'*'D')+
('B'*'C'*'D'))<->('A'*'B'*'C'*'D') [personal_id, name,
age, dept_no ].
074 001-3S:Host:012:"((SELECT PERSONAL_NO,
NAME, AGE, DEPT_CODE TECH_HISTORY.EMP_
NO FROM EMP, TECH_HISTORY WHERE AGE≧30
AND AGE<35 AND YEARS>12 AND TECH_HISTORY-
.SYSTEM='UNIX' EMP.PERSONAL_NO=TECH_HIS-
TORY.EMP_NO)
UNION
(SELECT PERSONAL_NO, NAME, AGE, DEPT_
CODE, TECH_HISTORY.EMP_NO FROM EMP,
TECH_HISTORY WHERE AGE≧30 AND AGE<35 AND
PROJ_CODE IS NULL AND TECH_HISTORY.SYS-
TEM='UNIX' EMP.PERSONAL_NO=TECH_HISTO-
RY.EMP_NO)
UNION
(SELECT PERSONAL_NO, NAME, AGE, DEPT_
CODE, TECH_HISTORY.EMP_NO FROM EMP,
TECH_HISTORY WHERE YEARS>12 AND PROJ_
CODE IS NULL AND TECH_HISTORY.SYSTEM=
'UNIX' EMP.PERSONAL_NO=TECH_HISTO-
RY.EMP_NO))
MINUS
(SELECT PERSONAL_NO, NAME, AGE, DEPT_
CODE, TECH_HISTORY.EMP_NO FROM EMP,
TECH_HISTORY WHERE AGE≧30 AND AGE<35 AND
YEARS>12 AND PROJ_CODE IS NULL AND TECH_
HISTORY.SYSTEM='UNIX' EMP.PERSONAL_NO=
TECH_HISTORY.EMP_NO);
075 001-3R:Host:012:"ready for output"
```

As explained in the above example, "the database retrieval method through set graph" realizes retrieval processing with the use of the graphical user interface to select and point the graphic objects, except the basic conditional formula of the set objects described through the phased operational flow (FIG. 15) which follows the logical way of human thinking.

The present invention brings about the below effects, by way of displaying the sets projected in the geometric rectangular shapes,

(a) to evaluate and recognize visually the quantity of the sets and logical relation between the sets from the size and layout of the geometric rectangular shapes,

(b) to realize data retrieval manipulation by creating 255 different retrieval conditional formulae, with the instruction of selecting combinations of the sets from eight logical sets, including a maximum of seven regional portions and the surrounding mother set, which are drawn with three rectangular regions, and

(c) to use the set graphs for comparing the sets with the time scale as well as for better explanation and presentation, because the quantity and relation of the abstract existence of the sets are recorded and shown as graphs.

In the database retrieval, such as information and documentation retrieval, the object of retrieval is not always clearly grasped, so under an uncertain and unclear tentative condition, retrieval processing is normally repeated on a trial and error basis. In such a retrieval processing, groping for the object itself under various retrieval conditions is meaningful, and retrieval is executed with a comparatively simple formula, and the sets are created and the object of retrieval is understood from the logical relation of those sets manipulated with a combination of the created sets using the boolean operators.

Furthermore, in case of the database retrieval of the personnel or management information, the best condition for retrieval can be obtained through describing complex conditions against multiple database files and make combinations of the sets obtained from the assumed multiple conditional values.

Such type of retrieval is very important and often found difficult to achieve because the logical challenge to understand the logical relation between the sets is more difficult than the technical challenge in computer manipulation. Therefore, it is still often observed that the user himself draws circles to show the sets on a sheet of paper to try to understand the relation of the sets from the drawing, and that after such manipulation, he can finally return to the keyboard of the computert to enter the retrieval conditions.

The present invention provides an integrated solution for the technical problem in the course of operation as well as for the logical problem in the human thinking, in highly complex retrieval processing or uncertain and unclear retrieval processing.

What is claimed is:

1. A database retrieval method comprising the steps of:
   a) selecting a first space object database;
   b) creating a plurality of set objects for retrieval of data from said space object database;
   c) displaying a visual representation of said plurality of set objects, wherein said display gives overlapping areas of said visual representation when data in said set objects is the same;
   d) selecting one or more areas of said display; and
   e) extracting data from said space object which is within said one or more areas selected.

2. The retrieval method of claim 1 wherein said plurality of sets does not exceed three.

3. The retrieval method of claim 1 further comprising the step of displaying a number of hit elements within one or more areas of said display.

4. The retrieval method of claim 1 further comprising the steps of:
   pre-registering set object definitions, and
   selecting at least one of said created plurality of set objects by selecting said pre-registered set object definitions.

5. The retrieval method in accordance with claim 2 wherein a count of each of said created set objects is displayed on a screen of a graphical display unit.

6. The retrieval method in accordance with claim 1 wherein said visual representation display shows a logical relationship among the created set objects.

7. The retrieval method in accordance with claim 6 wherein said visual representation display is a geometric rectangular region.

8. The retrieval method in accordance with claim 6 further comprising the step of displaying a logical statement of the relationship of the created set objects.

9. The retrieval method in accordance with claim 1 further comprising the steps of:
   creating a first data base set object from said extracted data of said first data base;
   selecting a second space object data base;
   creating a second data base set object which is defined by one or more set objects selected from said second data base;

creating a first visual display of said first data base set object;

creating a second visual display of said second data base set object;

displaying a visual representation of said first visual display of said first data base set object and of said second visual display of said second data base set object; and

extracting data from said first and second data bases which is within one or more areas selected from said first visual display of said first data base set object and of said second visual display of said second data base set object.

10. The retrieval method in accordance with claim **1** wherein said visual representation is drawn in a mode selected from the group consisting of rate, fit, and zoom.

11. A database retrieval apparatus comprising:

a) a means for selecting a space object database;

b) a means for creating a plurality of set objects for retrieval of data from said space object database;

c) a means for displaying a visual representation of said plurality of set objects, wherein said display gives overlapping areas of said visual representation when data in said set objects is the same;

d) a means for selecting one or more area of said display; and

e) a means for extracting data from said space object which is within said one or more areas selected.

12. The retrieval apparatus of claim **11** wherein said plurality of sets does not exceed three.

13. The retrieval apparatus of claim **11** further comprising a means for displaying a number of hit elements within one or more areas of said display, and said space object.

14. The retrieval apparatus of claim **11** further comprising:

a means for pre-registering set objects, and

a means for selecting at least one of said created set of objects by selecting said pre-registered set object.

15. The retrieval apparatus in accordance with claim **11** wherein a count of said created set objects is displayed on a screen of a graphical display unit.

16. The retrieval apparatus in accordance with claim **11** wherein said display shows a logical relationship among the created set objects.

17. The retrieval apparatus in accordance with claim **16** wherein said display is a geometric rectangular region.

18. The retrieval apparatus in accordance with claim **16** further comprising a means for displaying a logical statement of the relationship of the created set objects.

19. The retrieval apparatus in accordance with claim **11** further comprising:

a means for creating another set object from a space object database;

a means for creating another set object which is defined by the one or more areas selected when logically combined;

a means for displaying a visual display of said another set object obtained from a space object database;

a means for displaying a visual display of said object which is defined by one or more areas selected when logically combined;

a means for selecting one or more areas of said second visual display; and

a means for extracting data from said space object which is within one or more areas selected from said second visual display.

20. The retrieval apparatus according to claim **11** wherein said visual representation is drawn in a mode selected from the group consisting of rate, fit, and zoom.

* * * * *

46/3,K/57 (Item 57 from file: 350) **Links**
Derwent WPIX

015213993 **Image available**
WPI Acc No: 2003-274530/200327
XRPX Acc No: N03-217808
Object oriented interface provision method using internet for relational database, involves presenting object model mapped onto database schema in user -friendly graphical user interface to users

Patent Assignee: AT & T CORP (AMTT)
Inventor: NESHATFAR S; WARTY P
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6490581	B1	20021203	US 2000577103	A	20000524	200327	B

Priority Applications (No Type Date): US 2000577103 A 20000524
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6490581	B1		15	G06F-017/30	

Object oriented interface provision method using internet for relational database, involves presenting object model mapped onto database schema in user -friendly graphical user interface to users

Abstract (Basic):
... An **object model** mapped onto a database schema, is presented in a **user**-friendly **graphical user interface**. The graphical display is **modified** to prompt a **user** for setting an **object** constraint, in response to the **selections** of presented **objects**. A query is generated using the **set object** constraint.
... For providing **object** oriented interface to **users** in **relational** database management system, using internet...

....As the **object model** mapped onto the database schema is presented in a **user**-friendly **graphical user interface**, allows **users** with no knowledge of the database **query language** to perform sophisticated queries with just a few mouse clicks...

...The figure shows the **graphical user interface** for presenting information...
Title Terms: **OBJECT**;

International Patent Class (Main): **G06F-017/30**
Manual Codes (EPI/S-X): **T01-J05B3**...

...**T01-J05B4B**...

...**T01-J12A**...

...**T01-N01**

US006490581B1

(12) **United States Patent**
Neshatfar et al.

(10) **Patent No.: US 6,490,581 B1**
(45) **Date of Patent: Dec. 3, 2002**

(54) **SYSTEM AND METHOD FOR PROVIDING AN OBJECT-ORIENTED INTERFACE TO A RELATIONAL DATABASE**

(75) Inventors: **Shapour Neshatfar**, Manalapan, NJ (US); **Pramod Warty**, Freehold, NJ (US)

(73) Assignee: **AT&T Corp.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/577,103**

(22) Filed: **May 24, 2000**

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** ..... **707/4**; 707/103 R
(58) **Field of Search** ..... 707/2, 3, 4, 10, 707/103 R, 103 Z

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,586,311 A	*	12/1996	Davies et al.	345/764
5,627,979 A	*	5/1997	Chang et al.	345/763
5,809,266 A	*	9/1998	Touma et al.	345/764
6,016,488 A	*	1/2000	Bosworth et al.	707/4
6,122,641 A	*	9/2000	Williamson et al.	707/101
6,269,475 B1	*	7/2001	Farrell et al.	717/113

OTHER PUBLICATIONS

Keramopoulos et al, "The Users View Level of the GOQL Graphical Query Language" Jul. 1999, Proceedings 1999 IEEE International Conference on Information Visualization, pp. 82–86.*

de Carvalho et al, "A Visual Query System Implementing a Temporal Object–Oriented Model with Roles on a Relational Database" Nov. 1997, Proceedings XVII Inernational Confernce of the Chilean Computer Science Society, pp. 38–40.*

Epstein, "A Graphical Query Langueage for Object–Oriented Data Models," Oct. 1990, Proceedings of the 1990 IEEE Workshop on Visual Languages, pp. 36–41.*

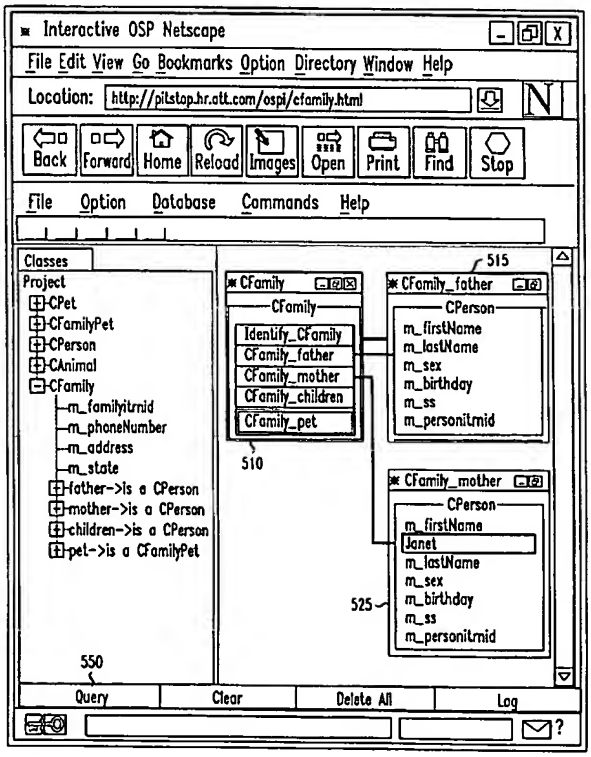
* cited by examiner

Primary Examiner—Jack Choules

(57) **ABSTRACT**

A method provides for an easily understood representation of the elements and relationships captured in a relational database. In accordance with the method an object model specification file and a database schema file are parsed to create a user-friendly graphical representation of object models and their relationship. The user is able perform a query by selecting one or more objects and setting constraints using the graphical representation of object models.

6 Claims, 9 Drawing Sheets



own collection of favorite books, the association of books to a child is easier to visualize when children are shown one at a time with a button labeled favorite Books, where clicking on that button, reveals the books.

Queries can also be made over multiple objects, related or otherwise. Consider the following query:

List pair of families with the same father's name (first name).

The above query can be performed by instantiating (clicking) the CFamily object twice. Expanding the father object on both, and drag the m_firstName from one father and drop it to another one and click the Query button as illustrated in FIG. 7.

Conclusion

The present invention takes advantage of the mapping performed by the OSP tool to present an easily understood description of the relationship between elements in a relational database. Furthermore, this new tool, OSPI, permits casual users to generate queries for complex databases.

What is claimed is:

1. A method of facilitating user operations on a relational database, comprising:

mapping an object model onto a database schema;

presenting the object model in a graphical user interface;

responsive to selections of presented objects, modifying said display to prompt a user to set an object constraint; and

generating a query using said constraint.

2. The method of claim 1 further comprising:

generating a web page based on the results of mapping said object model onto said database schema; and

said step of presenting includes providing said generated web page to a user.

3. The method of claim 1 wherein said step of presenting includes delineating in said graphical user graphical user interface,

a plurality of objects and relationships between the objects in said plurality of objects.

4. A method for presenting relational database information, comprising:

parsing an object model specification file and a database schema file to create a parse tree;

constructing a graphical representation of an object model using said parse tree;

transmitting said graphical representation to a user for display on a graphical user interface.

5. The method of claim 4 wherein said step of constructing comprises generating a web page representation of a plurality of objects and relationships among objects in said plurality.

6. The method of claim 5 further comprising:

receiving an object model selection;

presenting a list of objects corresponding to the selected object model;

receiving a selection of an object selected from said list of objects; and

transmitting constraint parameter information in response to said selection of an object.

* * * * *

46/3,K/75 (Item 75 from file: 350) **Links**
Derwent WPIX

014213980 **Image available**
WPI Acc No: 2002-034678/200204
XRPX Acc No: N02-026663

Database records relationship display method for relational database browsing, involves adding list of possible access pathways between related-recorded list based on user input and other records in database

Patent Assignee: ORILLION CORP (ORIL-N)
Inventor: LINDSEY T P
Number of Countries: 095 Number of Patents: 005
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
WO 200186504	A2	20011115	WO 2001US14581	A	20010504	200204	B
JP 2001318816	A	20011116	JP 2000314380	A	20001013	200208	
AU 200161215	A	20011120	AU 200161215	A	20010504	200219	
EP 1407382	A2	20040414	EP 2001935091	A	20010504	200426	
			WO 2001US14581	A	20010504		
AU 2001261215	A8	20051020	AU 2001261215	A	20010504	200615	

Priority Applications (No Type Date): US 2000565857 A 20000505
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 200186504	A2	E	27	G06F-017/30	
Designated States (National): AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG UZ VN YU ZA ZW					
Designated States (Regional): AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW MZ NL OA PT SD SE SL SZ TR TZ UG ZW					
JP 2001318816	A		19	G06F-012/00	
AU 200161215	A				Based on patent WO 200186504
EP 1407382	A2	E		G06F-017/30	Based on patent WO 200186504
Designated States (Regional): AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR					
AU 2001261215	A8			G06F-017/30	Based on patent WO 200186504

Database records relationship display method for relational database browsing, involves adding list of possible access pathways between related-recorded list based on user input and other records in database

Abstract (Basic):
... A list of possible **relationships** defining access pathways between a **selected** record and other records in a database, are displayed. A list of records **related** to **selected** records are added, based on **user** input. A list of possible

relationships defining pathways between the **related** record list and other records are added to the display.

... a) Computer-readable medium storing record **relationship** display program...

...b) Computer programmed to execute the database records **relationship** display program...

...For browsing **relational** database e.g. Oracle, Sybase, Microsoft Access, using standard **query language** (**SQL**).

...

...The data structures of different **relational** databases are viewed in a browsable format, easily by **choosing** any record as the initial point. The records within a database are easily verified and the **corresponding fields** or tables of various **relational** databases are renamed by the **user.**

...

...The figure shows multiple **relational** databases

...Title Terms: **RELATED;**

International Patent Class (Main): **G06F-012/00**...

...**G06F-017/30**

Manual Codes (EPI/S-X): **T01-J05B3**...

...**T01-J05B4B**...

...**T01-J12B**...

...**T01-S03**

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 November 2001 (15.11.2001)

PCT

(10) International Publication Number
WO 01/86504 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US01/14581

(22) International Filing Date: 4 May 2001 (04.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/565,857 5 May 2000 (05.05.2000) US

(71) Applicant: ORILLION CORPORATION [US/US]; Suite 110, 6925 Portwest, Houston, TX 77024 (US).

(72) Inventor: LINDSEY, Terry, P.; 17 Starviolet, The Woodlands, TX 77380 (US).

(74) Agent: CHICHESTER, Ronald, L.; Baker Botts L.L.P., One Shell Plaza, 910 Louisiana, Houston, TX 77002 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i)) for all designations*

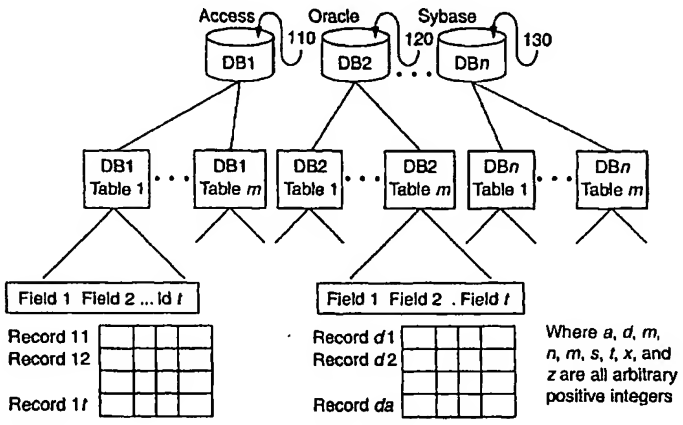
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for all designations*

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: BROWSER ORIENTED METHOD TO VIEW CONTENTS OF A PLURALITY OF DATABASES



(57) Abstract: A machine-executed method and apparatus for displaying relationships between records in one or more databases in a browseable format is disclosed. Each database comprises one or more tables. Each table includes one or more fields and one or more records. The method entails: (a) displaying, on a display, a field value of a selected record in a table of a selected database; (b) displaying on the display a list of possible relationships between the selected record and other records in the database, where each possible relationship defines a pathway from the selected record to a list of other records; (c) receiving a user input signal designating one of the possible relationships as a selected relationship; (d) adding to the display at least a portion of a list of records that are related to the selected record through the selected relationship, referred to as the related-records list; and (e) adding to the display a list of possible relationships between the related-records list and other records in the database, where each possible relationship defines a pathway from the related-records list to a list of other records.

WO 01/86504 A2

WHAT IS CLAIMED IS:

1. A method of displaying relationships between records in a database, the database comprising a plurality of tables, each table containing one or more records divided into one or more fields, the method comprising:

(a) determining the respective structures of the plurality of relational databases;

(b) displaying, on a display, a field value of a selected record in a table of the database;

(c) displaying on the display a list of possible relationships between the selected record and other records in the database, where each possible relationship defines a pathway from the selected record to a list of other records;

(d) receiving a user input signal designating one of the possible relationships as a selected relationship;

(e) adding to the display at least a portion of a list of records that are related to the selected record through the selected relationship, referred to as the related-records list; and

(f) adding to the display a list of possible relationships between the related-records list and other records in the database, where each possible relationship defines a pathway from the related-records list to a list of other records.

2. The method of claim 1, further comprising:

(g) receiving a user input signal designating a record in the related-records list as a new selected record; and

(h) repeating, for the new selected record, the operations recited in subparagraphs (d) through (f) of claim 1.

3. A method of displaying relationships between records of a plurality of relational databases of arbitrary structure, each database comprising a plurality of tables, each table containing one or more records divided into one or more fields, each table having a name, and each field having a name, the method comprising:

(a) determining the respective structures of the plurality of relational databases;

(b) integrating the respective structures of the plurality of relational databases into a front-end integrated data structure;

(c) displaying, on a display, a field value of a selected record in one of the plurality of tables in the plurality of relational databases;

(d) displaying on the display a list of possible relationships between the selected record and other records in the plurality of relational databases, where each possible relationship defines a pathway from the selected record to a list of other records;

(e) receiving a user input signal designating one of the possible relationships as a selected relationship;

(f) adding to the display at least a portion of a list of records that are related to the selected record through the selected relationship, referred to as the related-records list; and

(g) adding to the display a list of possible relationships between the related-records list and other records in the plurality of relational databases, where each possible relationship defines a pathway from the related-records list to a list of other records.

4. The method of claim 3, further comprising:

(a) receiving a user input signal designating a record in the related-records list as a new selected record; and

(b) repeating, for the new selected record, the operations recited in subparagraphs (e) through (g) of claim 3.

5. The method of claim 3, wherein integrating the respective structures comprises:

removing the conflicting names of tables and fields in the front-end integrated data structure; and

performing a join operation to establish a link among selected fields in at least two respective tables.

6. The method of claim 5, wherein removing the conflicting names comprises:

displaying, for a selected name, a cross-reference list of uses of the selected name in the plurality of relational databases;

receiving at least one editing command from the user, the at least one editing command designating an operation to be performed on a selected table or on a selected field; and

executing the at least one editing command.

7. The method of claim 6, wherein the at least one editing command comprises: receiving an alias designator which designates an alias for a selected name; and displaying the alias in lieu of the selected name.

8. The method of claim 7, further comprising: storing a representation of the relationship between the alias and the selected name.

9. A computer-readable medium encoded with instructions that, when executed by a computer, perform a method in accordance with a specified one of claims 1 through 8.

10. A computer programmed to perform a method in accordance with a specified one of claims 1 through 8.

46/3,K/119 (Item 119 from file: 350) **Links**
Derwent WPIX

010443433 **Image available**
WPI Acc No: 1995-344752/199544
XRPX Acc No: N95-257636

Database query system for forming semantically correct queries - uses query expert system to monitor structure of query and prevent user from building semantically incorrect queries

Patent Assignee: SOFTWARE AG (SOFT-N); SPEEDWARE LTEE (SPEE-N)
Inventor: SHWARTZ S P; SCHWARTZ S P
Number of Countries: 062 Number of Patents: 010
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
WO 9526003	A1	19950928	WO 95IB517	A	19950323	199544	B
AU 9526816	A	19951009	AU 9526816	A	19950323	199603	
US 5584024	A	19961210	US 94217099	A	19940324	199704	
EP 803100	A1	19971029	EP 95921945	A	19950323	199748	
			WO 95IB517	A	19950323		
JP 9510565	W	19971021	JP 95524526	A	19950323	199801	
			WO 95IB517	A	19950323		
US 5812840	A	19980922	US 94217099	A	19940324	199845	
			US 96723962	A	19960926		
MX 9604236	A1	19971201	MX 964236	A	19960923	199936	
EP 803100	B1	19991222	EP 95921945	A	19950323	200004	
			WO 95IB517	A	19950323		
DE 69514123	E	20000127	DE 614123	A	19950323	200012	
			EP 95921945	A	19950323		
			WO 95IB517	A	19950323		
CA 2186345	C	20000606	CA 2186345	A	19950323	200041	
			WO 95IB517	A	19950323		

Priority Applications (No Type Date): US 94217099 A 19940324; US 96723962 A 19960926
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 9526003	A1	E	102	G06F-017/30	
Designated States (National): AM AT AU BB BG BR BY CA CH CN CZ DE DK EE ES FI GB GE HU JP KE KG KP KR KZ LK LR LT LU LV MD MG MN MW MX NL NO NZ PL PT RO RU SD SE SI SK TJ TT UA US UZ VN					
Designated States (Regional): AT BE CH DE DK ES FR GB GR IE IT KE LU MC MW NL OA PT SD SE SZ UG					
AU 9526816	A			G06F-017/30	Based on patent WO 9526003
US 5584024	A		63	G06F-017/30	
EP 803100	A1	E		G06F-017/30	Based on patent WO 9526003
Designated States (Regional): AT BE CH DE DK ES FR GB IE IT LI NL PT SE					
JP 9510565	W		124	G06F-017/30	Based on patent WO 9526003
US 5812840	A			G06F-017/30	Cont of application US 94217099
					Cont of patent US 5584024
MX 9604236	A1			G06F-017/30	

EP 803100	B1	E	G06F-017/30	Based on patent WO 9526003
Designated States (Regional): AT BE CH DE DK ES FR GB IE IT LI NL PT SE				
DE 69514123	E		G06F-017/30	Based on patent EP 803100
				Based on patent WO 9526003
CA 2186345	C	E	G06F-017/30	Based on patent WO 9526003

... uses query expert system to monitor structure of query and prevent user from building semantically incorrect queries

...Abstract (Basic): The system includes a conceptual layer **manager** for storing conceptual information about the database including a predetermined structure and a query assistant which provides the **user** with a **set** of permissible **selections** from which to **build** a semantically correct database query for the database in an intermediate **query language**. A query generator receives a query in the intermediate **query language** from the query assistant and **converts** the query into the target **query language**.

...

...The **user** enters a query through a **set** of dialogue boxes in an intermediate language. A query expert system monitors the query as it is built, and using the information about the structure of the database, prevents the **user** from **building** semantically incorrect queries by disallowing **choices** in the dialogue box which would create incorrect queries...

...USE/ADVANTAGE - Guiding **user** to interactively create correct queries using structured **query language, SQL**. Permits **user** to enter only queries that are both syntactically and semantically valid

...Abstract (Equivalent): A database query system for interactively creating, with a **user**, a syntactically and semantically correct query for a **relational** database having a plurality of tables, each of said tables having a plurality of columns and having a predetermined **relationship** to another of said tables, said system comprising...

...a conceptual layer **manager** for storing conceptual information about the **relational** database, said conceptual information including structural information concerning the identity of each of the tables and columns and the directionality and cardinality of the **relationships** between the tables...

...a query assistant **user interface** (''QAUI'') presenting to the **user** a **selectable** table **set** of **selectable** tables from among the tables in the database, a **selectable** column **set** of **selectable** columns from among the columns of each of said tables in the database, and a **selectable** column operations **set** of **selectable** column operations on the columns, from which the **user** may

select tables, columns, and column operations to construct a database query for said database in an intermediate **query language**, said QAUI further accepting from the **user selections** of tables, columns, and column operations...

...QAUI to receive from said QAUI the identity of each table, column, or column operation **selected** by the **user**, said QAES returning to the QAUI after each **selection** by the **user** an **updated** version of said **selectable** table **set**, said **selectable** column **set**, and said **selectable** column operations **set**, said QAES excluding from said **selectable sets** any table, column, or column operation which, if **selected** by the **user**, would, based on the then-current state of the database query and said conceptual information...

...Title Terms: **USER**;

International Patent Class (Main): **G06F-017/30**

International Patent Class (Additional): **G06F-012/00**...

...**G06F-017/27**

Manual Codes (EPI/S-X): **T01-J05B3**...

...**T01-J16A**

US005812840A

United States Patent [19]

Shwartz

[11] Patent Number: 5,812,840

[45] Date of Patent: *Sep. 22, 1998

[54] **DATABASE QUERY SYSTEM**

[75] Inventor: **Steven P. Shwartz**, Orange, Conn.

[73] Assignee: **Speedware Ltee./Ltd.**, Toronto, Canada

[*] Notice: The term of this patent shall not extend beyond the expiration date of Pat. No. 5,584,024.

[21] Appl. No.: **723,962**

[22] Filed: **Sep. 26, 1996**

Related U.S. Application Data

[63] Continuation of Ser. No. 217,099, Mar. 24, 1994, Pat. No. 5,584,024.

[51] **Int. Cl.**[6] **G06F 17/30**

[52] **U.S. Cl.** **395/604**; 395/757; 395/922

[58] **Field of Search** 395/601, 604, 395/752, 757, 922

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,506,326	3/1985	Shaw et al.	364/300
4,688,195	8/1987	Thompson et al.	364/300
4,689,737	8/1987	Grant	364/200
4,736,296	4/1988	Katayama et al.	364/419
4,811,207	3/1989	Hikita et al.	364/200
4,829,423	5/1989	Tennant et al.	364/200
4,839,853	6/1989	Deerwester et al.	364/900
4,914,590	4/1990	Loatman et al.	364/419
4,930,071	5/1990	Tou et al.	364/300
4,931,935	6/1990	Ohira et al.	364/419
4,943,933	7/1990	Miyamoto et al.	364/513
4,974,191	11/1990	Amirghodsi et al.	364/900
4,994,967	2/1991	Asakawa	364/419
5,099,426	3/1992	Carlgren et al.	364/419
5,175,814	12/1992	Anick et al.	395/161
5,197,005	3/1993	Shwartz et al.	395/600
5,204,947	4/1993	Bernstein et al.	395/157
5,237,502	8/1993	White et al.	364/419
5,255,386	10/1993	Prager	395/600
5,265,014	11/1993	Haddock et al.	364/419

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0287310	10/1988	European Pat. Off. .
0387226	9/1990	European Pat. Off. .
63-219034	9/1988	Japan .

OTHER PUBLICATIONS

Wu, "A Knowledge–Based Database Assistant With A Menu Based Natural Language User–Interface," Oct. 10, 1993, *IEICI: Trans. Inf. & Syst.* V. E76–D N. 10, pp. 1276–1287.

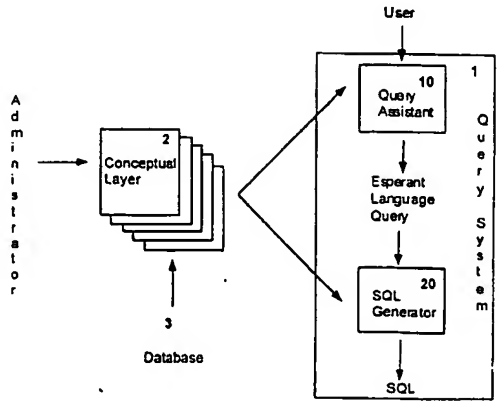
(List continued on next page.)

Primary Examiner—Thomas G. Black
Assistant Examiner—Jack M. Choules
Attorney, Agent, or Firm—Morgan, Lewis & Bockius LLP; C. Scott Talbot

[57] **ABSTRACT**

A database query system includes a query assistant that permits the user to enter only queries that are both syntactically and semantically valid (and that can be processed by an SQL generator to produce semantically valid SQL). Through the use of dialog boxes, a user enters a query in an intermediate English-like language which is easily understood by the user. A query expert system monitors the query as it is being built, and using information about the structure of the database, it prevents the user from building semantically incorrect queries by disallowing choices in the dialog boxes which would create incorrect queries. An SQL generator is also provided which uses a set of transformations and pattern substitutions to convert the intermediate language into a syntactically and semantically correct SQL query. The intermediate language can represent complex SQL queries while at the same time being easy to understand. The intermediate language is also designed to be easily converted into SQL queries. In addition to the query assistant and the SQL generator, an administrative facility is provided which allows an administrator to add a conceptual layer to the underlying database making it easier for the user to query the database. This conceptual layer may contain alternate names for columns and tables, paths specifying standard and complex joins, definitions for virtual tables and columns, and limitations on user access.

20 Claims, 26 Drawing Sheets



No new patterns are matched, and there is only one more pattern to match which, when applied, yields:

SHOW CUSTOMERS.NAME OFENTITY!! CUSTOMERS WHERE ORDERS >=1 PRODUCTS ORDERBY CUSTOMERS. CITY

Steps **408–412** are not applicable to this query, since there is no CREATE VIEW command needed for this type of query. If it were one of a specific set of queries which require CREATE VIEW SQL syntax, SQL Generator **20** would be called recursively to create the views. Since CREATE VIEW is not necessary, no new words or phrases for conversion were introduced.

In step **414**, the query is broken into SQL components. The query then becomes:

```
SELECT CUSTOMERS.NAME
WHERE ORDERS >= 1 PRODUCTS
ORDER BY CUSTOMERS.CITY
```

The LAST-ENTITY variable is set to CUSTOMERS, since the last table added to the select clause is from the table CUSTOMERS. The OFENTITY!! keyword introduced in the last pattern match is helpful in determining the LAST-ENTITY.

In steps **416–418**, the Where clause "ORDERS >=1 PRODUCTS" is applied to the patterns shown above, resulting in one match, with pattern **605**. By applying this pattern, the internal SQL structure for the query becomes:

```
SELECT CUSTOMERS.NAME
FROM CUSTOMERS T1
WHERE EXISTS
    SELECT *
    FROM PRODUCTS T2
    WHERE EXISTS
        SELECT *
        FROM ORDERS T3
        JOIN PRODUCTS T2
        JOIN CUSTOMERS T1
ORDER BY CUSTOMERS.CITY
```

The LAST-ENTITY variable is assigned the table PRODUCTS.

In step **420**, f there were any table names in the SELECT portion it would expand to include all of the tables columns. Also any virtual table would be expanded. Neither are present in this example, but are performed by simple substitution.

In step **422**, any columns in the SORT BY portion are added to SELECT if not present. This step converts the SELECT portion of the internal SQL to:

SELECT CUSTOMERS.CITY, CUSTOMERS.NAME

The date conversion function of step **424** is not applicable, since there are no dates in this example. Similarly, there are no virtual columns for expansion in step **426**.

If any aliases need to be specified to the FROM clause, they are made in step **428**. This query created the alias during the application of the Where rules, and no other tables were added to the from clause. The aliases are then substituted into the other sections as well. The internal SQL becomes:

```
SELECT T1.CITY T1.NAME
FROM CUSTOMERS TI
WHERE EXISTS
```

-continued

```
    SELECT *
    FROM PRODUCTS T2
    WHERE EXISTS
        SELECT *
        FROM ORDERS T3
        JOIN PRODUCTS T2
        JOIN CUSTOMERS T1
ORDER BY T1.CITY
```

In step **430**, the ORDER BY clause is converted to:

ORDER BY 1

In step **432**, required joins are computed from the internal SQL. They are represented here by the "JOIN Table Alias" statement, and indicates that those tables need to join the table listed in the FROM clause above it. From the prior discussion on join path calculations, the join paths created from the statements:

```
FROM ORDERS T3
JOIN PRODUCTS T2
JOIN CUSTOMERS T1
```

are [CUSTOMERS ORDERS] and [ORDERS LINE_ITEMS PRODUCTS].

Then, in steps **434** and **436**, since the join path calculation introduced a new table, LINE_ITEMS, the table needs to be added to the FROM clause with an alias to make:

FROM ORDERS T3, LINE_ITEMS T4

In step **438**, the joins are created and added to the where clause from the join paths and foreign key information in the conceptual layer to produce the following Where clause:

```
WHERE    T3.ORDER# = T4.ORDER#
    AND  T2.PRODUCT# = T4.PRODUCT#
    AND  T1.CUSTOMER# = T3.CUSTOMER#
```

Steps **440–444** are not applicable to this example. Finally, in step **446**, the internal SQL structure, which is now represented as:

```
SELECT T1.CITY T1.NAME
FROM CUSTOMERS T1
WHERE EXISTS
    SELECT *
    FROM PRODUCTS T2
    WHERE EXISTS
        SELECT *
        FROM ORDERS T3, LINE_ITEMS T4
        WHERE T3.ORDER# = T4.ORDER#
          AND T2.PRODUCT# = T4.PRODUCT#
              AND T1.CUSTOMER# = T3.CUSTOMER#
ORDER BY 1
```

is converted to textual SQL. The above representation of the internal SQL structure is in proper textual structure for a query. The process of conversion to the textual query from the internal structure is a trivial step of combining the clauses and running through a simple parser.

What is claimed is:

1. A database query system for interactively creating, with a user, a syntactically and semantically correct query for a relational database having a plurality of tables, each of said tables having a plurality of columns and having a predetermined relationship to another of said tables, said system comprising:

a conceptual layer manager storing conceptual information about the relational database, said conceptual

information including structural information concerning the identity of each of the tables and columns and the directionality and cardinality of the relationships between the tables;

a query assistant user interface ("QAUI") presenting to the user a selectable table set of selectable tables from among the tables in the database, a selectable column set of selectable columns from among the columns of each of said tables in the database, and a selectable column operations set of selectable column operations on the columns, from which the user may select tables, columns, and column operations to construct a database query for said database, said QAUI further accepting from the user selections of tables, columns, and column operations;

a query assistant expert ("QAES") coupled to said QAUI to receive from said QAUI the identity of each table, column, or column operation selected by the user, said QAES returning to the QAUI after each selection by the user an updated version of said selectable table set, said selectable column set, and said selectable column operations set, said QAES excluding from said selectable sets any table, column, or column operation which, if selected by the user, would, based on the then-current state of the database query and said conceptual information, produce a semantically incorrect query.

2. The system of claim 1 wherein said QAES includes:

a storage system for maintaining state information about the current state of a database query; and

a query expert logic system specifying to said QAUI said selectable sets by analyzing said state information maintained in said storage system and said conceptual information stored by said conceptual layer manager.

3. The system of claim 2 wherein the database includes at least three tables and if in the then-current state of said database query a first and second of the three tables are selected, the relationship between the first and second of the three tables is one-to-many, and the relationship between the first of the three tables and a third of the three tables is one-to-many, said query expert logic system excludes the third table from said selectable table set.

4. The system of claim 2 wherein said storage system includes:

a set of state variables; and

a set of access routines for adding, deleting, and modifying said state variables.

5. The system of claim 2 wherein said query expert logic system is composed of procedural logic.

6. The system of claim 2 wherein said query expert logic system is a rule-based expert system.

7. The system of claim 2 wherein said conceptual information further comprises one or more of the following: foreign keys, table join paths, table join expression for non-equijoins, virtual table definitions, virtual column definitions, table descriptions, column descriptions, hiddent tables, and hidden columns.

8. The system of claim 2 wherein said conceptual information further compriess table join expression for non-equijoins.

9. The system of claim 2 wherein if said current state of said database query includes an aggregate column operation on a colun in a first table, said query expert logic system excludes from said selectable table set any other of said tables that is more detailed than said first table or is joinable with said first table only through another more detailed table.

10. The system of claim 2 wherein said query expert logic system excludes from said selectable column set for any selected one of said tables any numeric column for which, if in said current state of said database query an aggregate column operation is applied to another column based on said selected table or based on another table having a one-to-one relationship with said selected table.

11. The system of claim 2 wherein said conceptual information further comprises virtual column definitions.

12. The system of claim 1 wherein each of said selectable table set, said selectable column set, and said selectable column operation set is mutually exclusive to a corresponding nonselectable table set, nonselectable column set, and nonselectable column operation set and is a subset of all tables, columns, and column operations, respectively, maintained by said conceptual layer manager which the user may next select in building a semantically correct database query.

13. The system of claim 12 wherein said QAUI displays said nonselectable sets and visually differentiates said selectable sets from said nonselectable sets.

14. The system of claim 1 wherein said database query is constructed in an intermediate query language and further comprising a query generator coupled to said QAUI to receive from said QAUI a completed database query in said intermediate query language, said query generator converting said query from said intermediate query language into a target query language, different from said intermediate query language, by a set of successive transformations by pattern substitution, said pattern substitutions including at lease one pattern that produces a correlated subquery when said target language is Structured Query Language.

15. A method for interactively building a syntactically and semantically correct query of a relational database from selections by a user, said database having a plurality of tables, each of said tables having a plurality of columns and having a predetermined relationship to another of said tables, said method comprising the steps of:

presenting to the user a selectable table set of selectable tables from among the tables in the database;

receiving from the user a selection of a first one of said selectable tables;

presenting to the user a selectable column set of selectable columns from among the columns based on said first selected table;

receiving from the user a selection of a first selected column from among said selectable columns'

presenting to the user a selectable column operation set of selectable column operations applicable to said first selected column;

receiving from the user a selection of one of said column operations;

determining from the selected table, column, and column operation and from information about the directionality and cardinality of the relationships between the tables the identity of unallowed tables, columns, or column operations which would, if selected by the user would produce a semantically incorrect query;

generating a first updated version of said selectable table set, a first updated version of said selectable column set, and a first updated version of said selectable column operation set, said sets excluding said unallowed tables, columns, and column operations, respectively; and

presenting to the user said first updated version of selectable table set.

16. The method of claim 15 wherein:

the database includes at least three tables; and

in said determining step if in the then-current state of said database query a first and second of the three tables are selected, the relationship between the first and second of the three tables is one-to-many, and the relationship between the first of the three tables and a third of the three tables is one-to-many, the third table is determined to be an unallowed table.

17. The method of claim **15** further comprising the steps of:

constructing from said user selections a database query in an intermediate query language;

converting said database query from said intermediate query language to a target query language, different from said intermediate query language.

18. The method of claim **17** wherein said target language is Structured Query Language and in said converting step said query is converted by a set of successive transformations by pattern substitution, said pattern substitutions including at lease one pattern that produces a correlated subquery.

19. The method of claim **15** wherein said selectable table set is mutually exclusive to a corresponding nonselectable table set and is a subset of all tables which the user may next select in building a semantically correct database query.

20. The method of claim **19** wherein said step of presenting to the user said first updated version of said selectable table set includes displaying said nonselectable sets and visually differentiating said selectable table set from said nonselectable table set.

* * * * *

US005584024A

United States Patent [19]

Shwartz

[11] Patent Number: 5,584,024

[45] Date of Patent: Dec. 10, 1996

[54] **INTERACTIVE DATABASE QUERY SYSTEM AND METHOD FOR PROHIBITING THE SELECTION OF SEMANTICALLY INCORRECT QUERY PARAMETERS**

[75] Inventor: **Steven P. Shwartz**, Orange, Conn.

[73] Assignee: **Software AG**, Germany

[21] Appl. No.: **217,099**

[22] Filed: **Mar. 24, 1994**

[51] **Int. Cl.**[6] **G06F 17/30**; G06F 17/27

[52] **U.S. Cl.** **395/604**; 395/922; 395/757; 364/274.2; 364/275.4; 364/283.3; 364/DIG. 1; 364/972.2; 364/974.6; 364/DIG. 2

[58] **Field of Search** 395/600, 922; 364/419.01, 419.07, 974.6, 972.2, 274.2, 274.7, 275.1, 275.4, 283.3, 282.1

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,506,326	3/1985	Shaw et al.	364/300
4,688,195	8/1987	Thompson et al.	364/300
4,689,737	8/1987	Grant	364/200
4,736,296	5/1988	Katayama et al.	364/419
4,811,207	3/1989	Hikita et al.	364/200
4,829,423	5/1989	Tennant et al.	364/200
4,839,853	6/1989	Deerwester et al.	364/900
4,914,590	3/1990	Loatman et al.	364/419
4,930,071	5/1990	Tou et al.	364/300
4,931,935	5/1990	Ohira et al.	364/419
4,943,933	7/1990	Miyamoto et al.	364/513
4,974,191	11/1990	Amirghodsi et al.	364/900
4,994,967	2/1991	Asakawa	364/419
5,099,426	3/1992	Carlgren et al.	364/419
5,175,814	12/1992	Anick et al.	395/161
5,197,005	3/1993	Shwartz et al.	395/600
5,204,947	4/1993	Bernstein et al.	395/157
5,237,502	8/1993	White et al.	364/419
5,255,386	10/1993	Prager	395/600
5,265,014	11/1993	Haddock et al.	364/419
5,265,065	11/1993	Turtle	395/600
5,349,526	9/1994	Potts, Sr. et al.	364/419.1
5,386,556	1/1995	Hedin et al.	395/600

FOREIGN PATENT DOCUMENTS

0287310	10/1988	European Pat. Off.	G06F 15/40
0387226	9/1990	European Pat. Off.	G06F 15/38
63-219034	9/1988	Japan	G06F 7/28

OTHER PUBLICATIONS

Wu, "A Knowledge-Based Database Assistant With A Menu Based Natural Language User-Interface" 10 Oct. 1993, IEICI: Trans. Inf. & Syst. V. E76-D N. 10 pp. 1276-1287.

(List continued on next page.)

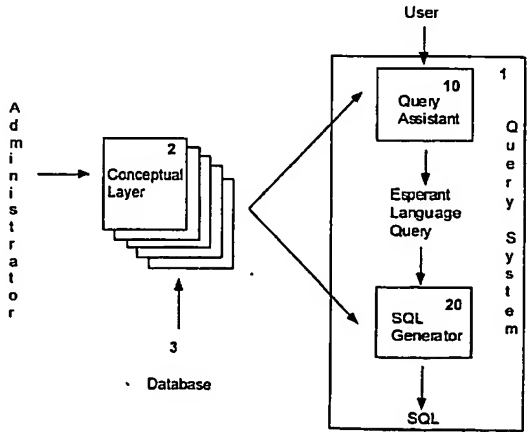
Primary Examiner—Wayne Amsbury
Assistant Examiner—Jack M. Choules
Attorney, Agent, or Firm—Howrey & Simon; C. Scott Talbot; Thomas G. Woolston

[57] **ABSTRACT**

A database query system includes a query assistant that permits the user to enter only queries that are both syntactically and semantically valid (and that can be processed by an SQL generator to produce semantically valid SQL). Through the use of dialog boxes, a user enters a query in an intermediate English-like language which is easily understood by the user. A query expert system monitors the query as it is being built, and using information about the structure of the database, it prevents the user from building semantically incorrect queries by disallowing choices in the dialog boxes which would create incorrect queries. An SQL generator is also provided which uses a set of transformations and pattern substitutions to convert the intermediate language into a syntactically and semantically correct SQL query.

The intermediate language can represent complex SQL queries while at the same time being easy to understand. The intermediate language is also designed to be easily converted into SQL queries. In addition to the query assistant and the SQL generator, an administrative facility is provided which allows an administrator to add a conceptual layer to the underlying database making it easier for the user to query the database. This conceptual layer may contain alternate names for columns and tables, paths specifying standard and complex joins, definitions for virtual tables and columns, and limitations on user access.

27 Claims, 26 Drawing Sheets



What is claimed is:

1. A database query system for interactively creating, with a user, a syntactically and semantically correct query for a relational database having a plurality of tables, each of said tables having a plurality of columns and having a predetermined relationship to another of said tables, said system comprising:

a conceptual layer manager for storing conceptual information about the relational database, said conceptual information including structural information concerning the identity of each of the tables and columns and the directionality and cardinality of the relationships between the tables;

a query assistant user interface ("QAUI") presenting to the user a selectable table set of selectable tables from among the tables in the database, a selectable column set of selectable columns from among the columns of each of said tables in the database, and a selectable column operations set of selectable column operations on the columns, from which the user may select tables, columns, and column operations to construct a database query for said database in an intermediate query language, said QAUI further accepting from the user selections of tables, columns, and column operations;

a query assistant expert ("QAES") coupled to said QAUI to receive from said QAUI the identity of each table, column, or column operation selected by the user, said QAES returning to the QAUI after each selection by the user an updated version of said selectable table set, said selectable column set, and said selectable column operations set, said QAES excluding from said selectable sets any table, column, or column operation which, if selected by the user, would, based on the then-current state of the database query and said conceptual information, produce a semantically incorrect query.

2. The database query system of claim 1 wherein said QAES includes:

a storage system for maintaining state information about the current state of a database query; and

a query expert logic system specifying to said QAUI said selectable sets by analyzing said state information maintained in said storage system and said conceptual information stored by said conceptual layer manager.

3. A database query system according to claim 2 wherein said storage system includes:

a set of state variables; and

a set of access routines for adding deleting and modifying said state variables.

4. A database query system according to claim 2 wherein said storage system includes:

a state database, said state database containing said state information; and

a set of database access routines for adding to, deleting from and modifying said state database.

5. A database query system according to claim 2 wherein said query expert logic system is composed of procedural logic.

6. A database query system according to claim 2 wherein said query expert logic system is a rule-based expert system.

7. A database query system according to claim 2 wherein said conceptual information further comprise one or more of the following: foreign keys, table join paths, table join expression for non-equijoins, virtual table definitions, virtual column definitions, table descriptions, column descriptions, hidden tables, and hidden columns.

8. A database query system according to claim 2 wherein said conceptual information further comprises table join expression for non-equijoins.

9. The database query system of claim 2 wherein if said current state of said database query includes an aggregate column operation on a column in a first table, said query expert logic system excludes from said selectable table set any other of said tables that is more detailed than said first table or is joinable with said first table only through another more detailed table.

10. The system of claim 2 wherein the database includes at least four tables and wherein if in the then-current state of said database query two of the tables are selected, said query expert logic system excludes from said selectable table set any other of said tables that does not form in combination with said two selected tables a navigable set.

11. The system of claim 2 wherein said query expert logic system excludes from said selectable column set for any selected one of said tables any numeric column for which, if in said current state of said database query an aggregate column operation is applied to another column based on said selected table or based on another table having a one-to-one relationship with said selected table.

12. A database query system according to claim 2 wherein said conceptual information further comprises virtual column definitions.

13. A database query system according to claim 12 wherein said virtual column definition contains primary key and foreign key references to define a join operation.

14. A database query system according to claim 1 wherein said each of said selectable table set, said selectable column set, and said selectable column operation set is mutually exclusive to a corresponding nonselectable table set, nonselectable column set, and nonselectable column operation set and is a subset of all tables, columns, and column operations, respectively, maintained by said conceptual layer manager which the user may next select in building a semantically correct database query.

15. A database query system according to claim 14 wherein said QAUI indicates to the user said selectable sets set of permissible selections and not displaying said nonselectable sets.

16. A database query system according to claim 14 wherein said QAUI displays said nonselectable sets and visually differentiates said selectable sets from said nonselectable sets.

17. A database query system according to claim 16 wherein said QAUI visually differentiates said sets by color.

18. A database query system according to claim 16 wherein said QAUI visually differentiates said selectable and nonselectable sets by type characteristic.

19. The database query system of claim 1 further comprising a query generator coupled to said QAUI to receive from said QAUI a completed database query in said intermediate query language, said query generator converting said query from said intermediate query language into a target query language different from said intermediate query language.

20. A database query system according to claim 19 wherein said target query language is Structured Query Language (SQL).

21. A database query system according to claim 19 wherein said query generator converts said intermediate language query into said target language by a set of successive transformations.

22. A database query system according to claim 21 wherein at least one of said set of successive transformations is transformation by pattern substitution.

23. A database query system according to claim 21 wherein said set of transformations comprises:

a set of structural transformations;

a set of transformation to include inferred information; and

a set of transformations by pattern substitution.

24. A method for interactively building a syntactically and semantically correct query of a relational database from selections by a user, said database having a plurality of tables, each of said tables having a plurality of columns and having a predetermined relationship to another of said tables, said method comprising the steps of:

presenting to the user a selectable table set of selectable tables from among the tables in the database;

recciving from the user a selection of a first one of said selectable tables;

presenting to the user a selectable column set of selectable columns from among the columns based on said first selected table;

receiving from the user a selection of a first selected column from among said selectable columns;

presenting to the user a selectable column operation set of selectable column operations applicable to said first selected column;

receiving from the user a selection of one of said column operations;

presenting to the user a first updated version of said selectable table set from which the user may select a second selected table, said selectable table set excluding any table that is more detailed than said first selected table on which said selected column operation is applied or that is joinable with said first selected table only through a more detailed table if said selected column operation is an aggregate operation.

25. The method of claim 24 wherein said first updated version of said selectable table set further excludes any table that is not joinable with said first selected table.

26. The method of claim 25 wherein the database includes at least four tables and further comprising the steps of:

receiving from the user a selection of a second selected table from said first updated version of said selectable table set; and

presenting to the user a second updated version of said selectable table set from which the user may select a third selected table, said selectable table set excluding any table the selection of which does not form in combination with said first and second selected tables a navigable set.

27. The method of claim 26 further comprising the steps of:

receiving from the user a selection of a second selected table from said first updated version of said selectable table set; and

presenting to the user an updated version of said selectable column set based on said second selected table, said selectable columns set excluding any column that contains non-numeric information if said second selected table is the same as said first selected table or has a one-to-one relationship with said first selected table.

* * * * *

46/3,K/99 (Item 99 from file: 350) **Links**
Derwent WPIX

012238721 **Image available**
WPI Acc No: 1999-044829/199904
Related WPI Acc No: 2002-033185
XRPX Acc No: N99-032727

User interface creating method for constraint handling in product configurator software - involves using control palette for placing user control at predefined position on configurator screen display template and several parameters associated with user control

Patent Assignee: BT SQUARED TECHNOLOGIES INC (BTSQ-N)
Inventor: GELLER S D; HEYDA M S; NEES R
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 5844554	A	19981201	US 96718645	A	19960917	199904	B

Priority Applications (No Type Date): US 96718645 A 19960917
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5844554	A		79	G06F-015/00	

User interface creating method for constraint handling in product configurator software...

...**involves using control palette for placing user control at predefined position on configurator screen display template and several parameters associated with user control**

...Abstract (Basic): involves displaying a form creation region on the display. The form creation region includes a **configuration** screen display template for display of **user** controls and information in the **user** product **configuration** program. A control palette containing several control **objects** which upon activation, places and displays **corresponding user** control at predefined position on **configurator** screen display template. A developer command issued with respect to the **user** control causes a parameter **selector** window to be displayed...

...The window contains several parameters associated with the **user** control. One such parameter is **selected** and associated with the **user** control. The information relating to the **selected** parameter is retrieved from the memory where it is stored, and used **corresponding** to the **user** control for computing product **configurations**.

...

...ADVANTAGE - Allows external data tables to be **updated** independent of **configuration** software. Enables **updating** **configurator** software by merely executing **SQL** queries, avoiding rewriting of program code. Features very rapid execution time and quick response. Features **builder** windows and editors that allow tree-type hierarchical navigation and double-click **selection,** and well-organised queries

Title Terms: **USER;**

International Patent Class (Main): **G06F-015/00**

Manual Codes (EPI/S-X): **T01-J12**

US005844554A

United States Patent [19]

Geller et al.

[11] Patent Number: 5,844,554

[45] Date of Patent: Dec. 1, 1998

[54] METHODS AND SYSTEMS FOR USER INTERFACES AND CONSTRAINT HANDLING CONFIGURATIONS SOFTWARE

[75] Inventors: Scott D. Geller, Atlanta; Michael S. Heyda, Duluth; Robert Nees, Canton, all of Ga.

[73] Assignee: BT Squared Technologies, Inc., Atlanta, Ga.

[21] Appl. No.: 718,645

[22] Filed: Sep. 17, 1996

[51] Int. Cl.[6] G06F 15/00

[52] U.S. Cl. 345/333; 345/335; 345/347; 345/962; 345/967

[58] Field of Search 345/326, 333, 345/339, 342, 343, 347, 964, 968, 970, 335, 348, 962, 965; 705/1–4, 26, 29

[56] References Cited

U.S. PATENT DOCUMENTS

4,591,983	5/1986	Bennett et al.	395/65
4,688,195	8/1987	Thompson et al.	395/12
4,920,499	4/1990	Skeirik	395/12
4,992,940	2/1991	Dworkin	705/26
5,032,989	7/1991	Tornetta	705/1
5,212,634	5/1993	Washizaki et al.	364/400
5,228,116	7/1993	Harris et al.	395/54
5,260,866	11/1993	Lisinki et al.	705/29
5,267,146	11/1993	Shimizu et al.	364/512
5,307,260	4/1994	Watanabe et al.	395/500
5,307,261	4/1994	Maki et al.	705/29
5,311,424	5/1994	Mukherjee et al.	705/29
5,367,627	11/1994	Johnson	345/357
5,369,732	11/1994	Lynch et al.	395/51
5,446,653	8/1995	Miller et al.	705/4
5,471,596	11/1995	Brown, III	707/103
5,493,490	2/1996	Johnson	705/26
5,515,524	5/1996	Lynch et al.	395/500
5,523,942	6/1996	Tyler et al.	705/4

OTHER PUBLICATIONS

The advertising brochure "Trilogy PriceBuilder", Trilogy Development, Inc. The date of availability of this brochure and of the described item is not known, but is believed to be more than one year before the filing date of the present application.

The advertising brochure "SC Config", Trilogy Development, Inc. The date of availability of this brochure and of the described item is not known, but is believed to be more than one year before the filing date of the present application.

The advertising brochure "sellingPoint™", Concentra Corporation, Burlington, MA. Except for the copyright notice date of 1995 printed on the brochure, the date of availability of this brochure and of the described item is not known.

The advertising brochure "SalesLogic™", OBJIX Systems Development, Inc. Except for the copyright notice date of 1996 printed on the brochure, the date of availability of this brochure and of the described item is not known.

(List continued on next page.)

Primary Examiner—Huynh Ba

Attorney, Agent, or Firm—Jones & Askew, LLP

[57] ABSTRACT

A computer implemented method of generating a user product configuration program module from a development environment. The user product configuration program module includes user controls that allow user input of information for use in configuration computations. Methods are disclosed for creating and maintaining the logic for a configuration program module in the form of configuration parameters; creating and maintaining the visual controls and user interface; and linking created visual controls with underlying structure represented by the parameters. Parameters assume values and are indicated as valid or invalid through operation of constraints and queries. Further methods include display of parameter creation and selection windows, and query creation and selection windows for creating SQL queries to access data in external tables. Parameters, queries, and constraints can be displayed in expandable and collapsible hierarchies, and quickly utilized in formulas, queries, and logical expressions by clicking in an expanded hierarchical display. Dependencies occurring in the underlying configuration logic are evaluated in both the forward and reverse direction so as to provide for very fast execution of the resultant configuration program module when a user provides new data via a user control.

47 Claims, 49 Drawing Sheets

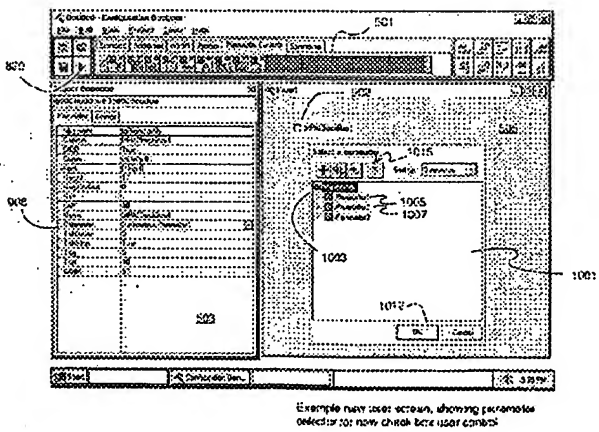


FIG. **45** illustrates the process **4500** executed within a compiled configuration program module to evaluate and apply constraints to a parameter, for purposes of determining whether constraints result in a valid or invalid condition for a parameter, as well as determine the values that a parameter can assume and the conditions for applying a constraint to parameter. It will be understood that the steps shown in FIG. **45** are executed within the compiled configuration program module each time that a user, not a developer, of the compiled configuration program module provides information input to the configuration program by activating a user control such as checking a check box, entering text into an edit box, activating a radio button, etc.

Starting at **4501**, the first inquiry made is whether there are one or more constraints associated with a particular parameter whose value is being evaluated in response to input of information by a user of the compiled configuration program module. If so, control passes to process **4503**. Process **4503** results in the execution of the conditions set forth in the "UsedWhen" property of the constraint. It will be recalled from previous discussion that the UsedWhen property, as shown at **1634** in FIG. **16**A, determines conditions for applying constraints. These properties are shown for the various types of constraints at **1634** for a query constraint (FIG. **16**A), **1734** for a discrete constraint (FIG. **17**), **1834** for a string list constraint (FIG. **18**), **2134** for a range constraint (FIG. **21**), and **2310** for a formula constraint (FIG. **23**). Further, it will be recalled from the discussion in conjunction with FIG. **20** how this conditional logic is employed by the UsedWhen property for application of a constraint to a parameter.

After process **4503**, the inquiry is made at decision **4505** whether the particular constraint is used. If not, control passes to decision **4509**, and the inquiry made whether there are more constraints associated with the particular parameter. It will be recalled that a number of constraints can be applied to a given parameter, such as the constraints **1622**, **1624**, **1626**, all of which are applied to the selected SoundSystem parameter **1608** in FIG. **16**A. If there are more constraints, control passes to step **4513**, the next constraint is evaluated, and control returns to process **4503**. If there are no more constraints at this point, the no branch is taken from decision **4509** and the process executes. In this case, a Valid condition for the parameter being evaluated is indicated, as no constraints have been applied that might cause the condition to be invalid.

Return new to step **4505** where the inquiry is made whether a constraint is used. If so, the yes branch is taken to process **4520**, and a validate process is executed to determine whether the application of the particular constraint validates the associated parameter or results in an invalid condition. This results from evaluation of any formulas or other computations in conjunction with the constraint.

At decision **4522**, if the constraint validates the parameter, the yes branch is taken to step **4525**, and the process exits with an indication of a valid condition. If not, the no branch is taken to decision **4530**, and the inquiry is made whether there are more constraints to be applied to this parameter. If not, the no branch is taken to step **4535**, and an invalid condition is indicated.

An invalid condition is indicated at this stage because it has been determined that a particular constraint is to be employed, but such constraint did not result in validating the associated parameter, and there are no more constraints that might further affect the validity or invalidity of the parameter. In this case, an invalid condition is indicated for the parameter.

However, if there are more constraints that might result in validation of the parameter, the yes branch is taken from decision **4530** to step **4537**. Here, the next constraint is evaluated, and at process **4539** the UsedWhen condition of the constraint is evaluated in a manner as described above. At decision **4541**, an inquiry is made whether the constraint being evaluated has been used. If not, the program loops back to **4530**. If at decision **4541** the constraint is used, control passes to step **4544**, and the Execute Validate process, similar to that of step **4520**, is executed. The next step is **4548**, and the inquiry made whether the constraint being evaluated validates the parameter. If so, the parameter assumes a valid condition at **4525** and the process exits. On the other hand, if not, control returns to step **4530** and further inquiries are made whether there are more constraints that might validate the parameter.

From the foregoing, those skilled in the art will appreciate that the preceding steps allow the very quick determination of the effect of the input of information via a user control in the executed configuration program module.

Finally, it will be understood that the preferred embodiment has been disclosed by way of example, and that other modifications may occur to those skilled in the art without departing from the scope and spirit of the appended claims.

What is claimed is:

1. A computer-implemented method of generating a user product configuration program module that executes product configuration computations, the user product configuration program module including user controls that allow user input of information for use in the product configuration computations, the method being carried out in a developer environment on a computer system including a display, a command entry device, a pointing device, and a memory, comprising the steps of:

displaying a form creation region on the display, the form creation region comprising a configurator screen display template for display of user controls and information in the user product configuration program;

displaying a controls palette comprising a plurality of selectable controls in a palette region on the display, each of the plurality of controls comprising a control object that, when activated by a command, causes placement and display of a user control corresponding to the activated control at a predetermined position on the configurator screen display template;

in response to a developer command in the developer environment with respect to the user control on the configurator screen display template, displaying a parameter selector window, the parameter selector window comprising a plurality of parameters that can be associated with the user control, information associated with the parameters being stored in the memory;

in response to a developer command in the developer environment with respect to a selected one of the plurality of parameters in the parameter selector window, associating the selected parameter with the user control,

whereby information stored in memory associated with the selected parameter, as influenced by a user command with respect to the user control as utilized in the user product configuration program, is utilized in product configuration computations.

2. The computer-implemented method of claim **1**, further comprising the step of displaying an object inspector region on the display comprising information corresponding to one or more predetermined properties of the user control, the

predetermined properties including a parameter property corresponding to information utilized in a product configuration computation; and

wherein the parameter selector window is displayed in response to selection of the parameter property of the user control in the object inspector region on the display by the developer.

3. The computer-implemented method of claim 1, further comprising the steps of:

in response to a compile command by the developer, compiling program code corresponding to the user control and information corresponding to the selected parameter into an executable configuration program module; and

providing the executable configuration program module as an output file, the executable configuration program module, when installed on a computer system, causing display of a user configurator screen display corresponding to the configurator screen display template and including the user control.

4. The computer-implemented method of claim 1, wherein the selected parameter is operative to assume an invalid state in response to predetermined conditions during execution of the configuration program module,

whereby during execution of the configuration program module, in response to user input of information via the user control that causes the predetermined condition, providing a visual indication of the invalid condition on the user configurator screen display.

5. The computer-implemented method of claim 1, wherein the parameter selector window includes a control that expands the display of parameters into a hierarchical arrangement of parameters.

6. The computer-implemented method of claim 5, wherein the parameter selector window includes a parameter group icon that when activated by a developer command, expands to display one or more parameters or other group icons associated with the parameter group icon.

7. The computer-implemented method of claim 1, wherein the control object in the controls palette is activated by the developer pointing to a predetermined region on the controls palette and activating a command entry device.

8. The computer-implemented method of claim 1, further comprising the steps of:

in response to a developer command in the developer environment, displaying a parameter explorer window on the display, the parameter explorer window comprising an editable display of the plurality of selectable parameters stored in the memory.

9. The method of claim 8, further comprising the step of allowing developer modification of a parameter property by direct data entry with the command entry device.

10. The computer-implemented method of claim 9, wherein the developer modification of the selected parameter property comprises use of a formula.

11. The computer-implemented method of claim 10, wherein the modification of the selected parameter property by use of a formula comprises the steps of:

in response to a developer command in the developer environment, displaying a parameter explorer window on the display, the parameter explorer window comprising an arrangement of the plurality of selectable parameters;

in response to selection of one of the parameters in the parameter explorer window, displaying the selected parameter's properties in an object inspector region on the display, the parameter properties including a formula property;

in response to selection of the formula property, displaying a formula builder window on the display;

allowing developer entry of a functional expression for assignment of a data value to the selected parameter.

12. A computer-implemented method for creating and maintaining configuration logic for a user product configuration program module that executes product configuration computations, the method being carried out in a developer environment on a computer system including a display, a command entry device, a pointing device, and a memory, comprising the steps of:

displaying a user control in the developer environment, the user control allowing user input of information for use in the configuration computations during execution of the user product configuration program module;

in response to a first developer command in the developer environment, displaying a parameter window on the display, the parameter window comprising a developer interface for creating and maintaining a plurality of selectable parameters stored in the memory;

in response to a second developer command in the developer environment with respect to a selected parameter in the parameter window, displaying a constraint item in association with the selected parameter, the constraint item being operative for constraining information that can be associated with the selected parameter; and

in response to a third developer command in the developer environment, associating a selected parameter with the user control,

whereby the user product configuration program module is operative during execution of the user product configuration program module to receive user input via the user control and apply information associated with the parameter associated with the user control in the product configuration computations.

13. The computer-implemented method of claim 12, wherein the constraint item comprises a query type constraint.

14. The computer-implemented method of claim 13, further comprising the steps of:

in response to a fourth developer command in the developer environment with respect to the query type constraint, displaying a query selector window on the display, the query selector window comprising a plurality of queries that can be associated with parameter information associated with the query being obtained from an external database and stored in the memory.

15. The computer-implemented method of claim 14, further comprising the steps of:

in response to a fifth developer command in the developer environment, displaying a query explorer window on the display, the query explorer window comprising a developer interface for creating and maintaining a plurality of selectable queries in the memory, each of the queries comprising a set of external database commands executable by the computer for retrieving information from an external database for association with one of the plurality of parameters.

16. The computer-implemented method of claim 15, wherein the query selector window and the query explorer window comprise separate windows.

17. The computer-implemented method of claim 12, wherein the constraint item is one or more selected from the

group comprising: a query constraint, a discrete constraint, a formula constraint, a string list constraint, and a range constraint.

18. The computer-implemented method of claim 12, further comprising the steps of:

displaying a parameter group icon in the parameter window, the parameter group icon being associated with an organizational concept with which a plurality of parameters can be associated; and

in response to a developer command with respect to the parameter group icon, expanding the display of parameters associated with the parameter group icon, thereby allowing creation of a hierarchically expandable and collapsible display of parameters.

19. The computer-implemented method of claim 12, wherein the parameter window is a parameter explorer window, and further comprising the steps of:

in response to a sixth developer command in the developer environment, displaying a parameter selector window on the display, the parameter selector window comprising a developer interface for associating a selected parameter stored in the memory with a selected user control.

20. The computer-implemented method of claim 19, wherein the parameter selector window is displayed in response to a developer command with respect to a control placed on a configuration screen display template, and further comprising the steps of:

displaying a form creation region on the display, the form creation region comprising a configurator screen display template for display of user controls and information in the user product configuration program;

displaying a controls palette comprising a plurality of selectable controls in a palette region on the display, each of the plurality of controls comprising a control object that, when activated by a command, causes placement and display of a user control corresponding to the activated control on the configurator screen display template; and

in response to a developer command with respect to a selected parameter in the parameter selector window, associating the selected parameter with the user control on the configurator screen display template,

whereby information stored in memory associated with the selected parameter, as influenced by a user command with respect to the user control as utilized in the user product configuration program, is utilized in product configuration computations.

21. The computer-implemented method of claim 20, further comprising the step of displaying an object inspector region on the display comprising information corresponding to one or more predetermined properties of the user control on the configurator screen display template, the predetermined properties including a parameter property corresponding to information utilized in a product configuration computation; and

wherein the parameter selector window is displayed in response to selection of the parameter property of the user control in the object inspector region on the display by the developer.

22. The computer-implemented method of claim 12, further comprising the steps of:

in response to a compile command by the developer, compiling program code corresponding to the user control and information corresponding to the selected parameter into an executable configuration program module; and

providing the executable configuration program module as an output file, the executable configuration program module, when installed on a computer system, causing display of a user configurator screen display including the user control.

23. The computer-implemented method of claim 12, wherein the selected parameter is operative to assume an invalid state in response to predetermined conditions during execution of the configuration program module,

whereby during execution of the configuration program module, in response to user input of information via the user control that causes the predetermined condition, providing a visual indication of the invalid condition on the user configurator screen display.

24. A method of generating a user product configuration program module that executes product configuration computations, the method being carried out in a developer environment on a computer system including a display, a command entry device, a pointing device, and a memory, comprising the steps of:

displaying a form creation region on the display, the form creation region comprising a configurator screen display template for display of user controls and information in the user product configuration program module;

displaying a controls palette comprising a plurality of selectable controls in a palette region on the display, each of the plurality of controls comprising a control object that, when activated by a command by a developer, causes placement and display of a user control corresponding to the activated control on the configurator screen display template;

displaying an object inspector region on the display comprising information corresponding to one or more predetermined properties of the user control, the predetermined properties including a parameter property corresponding to information utilized in a product configuration computation;

in response to selection of the parameter property of the user control in the object inspector region on the display by the developer, displaying a parameter selector window on the display, the parameter selector window comprising an arrangement of a plurality of selectable parameters;

in response to selection of one of the selectable parameters in the parameter selector window by the developer, associating information corresponding to the selected parameter with the user control;

in response to a compile command by the developer, compiling program code corresponding to the user control and information corresponding to the selected parameter into an executable configuration program module; and

providing the executable configuration program module as an output file, the executable configuration program module, when installed on a computer system, causing display of a user configurator screen display corresponding to the configurator screen display template and including the user control.

25. The method of claim 24, wherein the control control in the controls palette is activated by pointing to a predetermined region on the controls palette and activating the command entry device.

26. The method of claim 24, further comprising the step of allowing developer modification of the selected parameter property by direct data entry with the command entry device.

27. The method of claim 24, wherein the arrangement of the plurality of selectable parameters displayed in the parameter selector window comprises a hierarchical display.

28. The method of claim 24, further comprising the step of allowing developer modification of a parameter property of the user control on the configurator screen display template.

29. The method of claim 28, wherein the developer modification of the selected parameter property comprises use of a formula.

30. The method of claim 29, wherein the modification of the selected parameter property by use of a formula comprises the steps of:

in response to a developer command, displaying a parameter explorer window on the display, the parameter explorer window comprising an arrangement of the plurality of selectable parameters;

in response to selection of one of the parameters in the parameter explorer window, displaying the selected parameter's properties in the object inspector region on the display, the parameter properties including a formula property;

in response to selection of the formula property, displaying a formula builder window on the display;

allowing developer entry of a functional expression for assignment of a data value to the selected parameter.

31. The method of claim 30, wherein the formula builder window comprises a variables pane, a functions pane, and a formula pane; and

in response to selection by the developer of a function in the functions pane, utilizing the selected function as a part of a formula displayed in the formula pane.

32. The method of claim 31, wherein the functions in the functions pane are selected from the group comprising Assign, AssignWhen, AssignWhenElse, Lookup, Display, and DisplayWhen.

33. The method of claim 31, wherein one of the functions is a Lookup function, wherein an external data table including information for use in the function is stored in the memory, and further comprising the steps of:

displaying a function template in the formula pane, the function template comprising editable text string corresponding to a mathematical expression;

allowing developer editing of the function template in the formula pane;

displaying a hierarchical arrangement of parameters and queries in the variables pane;

in response to selection of a query in the variables pane with the pointing device, displaying the selected query in the formula pane; and

allowing developer editing of selected query.

34. A method of representing a constraint to a configuration parameter in the user interface of a configuration program module that executes product configuration computations, the method being carried out in a developer environment on a computer system including a display, a command entry device, and a memory, comprising the steps of:

storing a plurality of selectable parameters in a predetermined arrangement in the memory, each of the parameters comprising information utilized in a product configuration computation;

displaying on the display a configurator screen display template for display of user controls and information in the configuration program module;

displaying a user control on the screen display template, the user control being operative, in the configuration program module, for receiving user information input;

displaying an object inspector region on the display comprising information corresponding to properties of the user control displayed on the screen display template;

in response to developer selection of a parameters property of the user control in the object inspector region, displaying a first parameter window on the display for selection of a parameter, the first parameter window comprising a display of the plurality of selectable parameters stored in the memory;

in response to developer selection of one of the parameters in the first parameter window, associating the selected parameter with the user control in the memory;

in response to a first developer command, displaying a second parameter window on the display, the second parameter window comprising an editable display of the plurality of selectable parameters stored in the memory;

in response to a second developer command relating to a selected one of the parameters in the second parameter window, displaying a constraint item on the display in association with the selected parameter;

in response to a third developer command relating to the displayed constraint item, receiving developer input regarding the application of the constraint item to the selected parameter; and

in response to a developer compile command, providing the configuration program module as an output, the configuration program module being responsive, when run on a computer system, to information input by a user via the user control for determining application of the constraint to the selected parameter.

35. The method of claim 34, wherein the step of receiving developer input regarding application of the constraint item to the selected parameter comprises displaying properties of the constraint item in the object inspector region, the properties of the constraint item including a conditional application property, and

wherein in response to a developer command relating to the conditional application property, displaying a formula builder window on the display, the formula builder window comprising a region for developer input of a logical condition for application of the constraint represented by the constraint item to the parameter.

36. The method of claim 35, wherein the conditional application property is selected from the group comprising: UsedWhen, UsedWhenExcept, Display, DisplayWhen, and Lookup.

37. The method of claim 35, wherein the condition for application of the particular constraint represented by the constraint item to the parameter comprises a Boolean function that evaluates to true or false at run time, based on user input.

38. The method of claim 34, wherein the constraint item is selectable from the group comprising a discrete constraint, a range constraint, a string constraint, a query constraint, and a formula constraint.

39. The method of claim 34, wherein first parameter window for displaying the plurality of selectable parameters includes a parameter group icon, the parameter group icon comprising display for organizing a collection of parameters into a logical grouping on the display.

40. The method of claim 34, wherein the first parameter window and the second parameter window are separate windows on the display.

41. The method of claim 34, further comprising the steps of:

displaying a constraint list icon in association with the selected parameter in the second parameter window:

in response to developer selection of the constraint list icon in the parameter explorer window, displaying a constraint menu comprising a plurality of selectable constraint type controls;

in response to developer selection of one of the selectable constraint type controls, displaying a new constraint item in association with the selected parameter in the second parameter window.

42. The method of claim 34, further comprising the steps of:

in response to developer selection of a constraint item in the second parameter window, displaying properties of the selected constraint item in the object inspector region; and

allowing developer modification of the properties of the selected constraint item in the object inspector region.

43. The method of claim 34, further comprising the steps of:

receiving user input of information into a computer system running the configuration program module via the user control;

in response to determination that the information provided by the user input violates a constraint represented by a constraint item, providing a visual indication of the constraint violation on the user configurator screen display.

44. The method of claim 43, wherein the visual indication of the constraint violation comprises display of a red indication on a displayed user control associated with a parameter including the violated constraint.

45. The method of claim 44, wherein the visual indication of the constraint violation is provided on all user controls that are associated with the parameter having the violated constraint.

46. The method of claim 34, wherein the selected parameter is a first parameter whose value is dependent on the value of a second parameter,

wherein the assignment of a value to the first parameter comprises evaluation of a formula containing the second parameter,

wherein the value of the second parameter is a function of a user control, and

wherein information input by a user via the user control is reflected in real time to the first parameter.

47. The method of claim 34, wherein a plurality of constraint items may be associated with the selected parameter, and wherein the effect of the plural constraint items on the parameter is additive.

* * * * *

46/3,K/39 (Item 39 from file: 350) **Links**
Derwent WPIX

015624881 **Image available**
WPI Acc No: 2003-687052/200365
Related WPI Acc No: 2005-793721
XRPX Acc No: N03-548739

Machine-readable medium for managing functions that express business rules to allow calling, maintaining functions and providing an execution framework for functions

Patent Assignee: DAMAN INC (DAMA-N)
Inventor: GHATATE B
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6338069	B1	20020108	US 98210340	A	19981211	200365	B

Priority Applications (No Type Date): US 98210340 A 19981211
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6338069	B1		60	G06F-017/30	

Abstract (Basic):

Function **metadata'** is stored and allows the dynamic execution of a function e.g. execution of a command. When the engine receives a request e.g. for a **set** of **fields**, an execution plan can be created by determining functions to be processed based upon their inputs and outputs identified by the function **metadata.**

... An **object** technology infrastructure is formed to store data and **metadata** fro the functions to be maintained. **Metadata** about a function can include data describing what a function does, a 'cost' associated with...

...function and the input and output parameters required by the function. The exposure of the **metadata** regarding the functions' input and output parameters allows an engine to track input/output **relationship** between the functions and therefore define the order of execution...

...allow calling functions, maintaining functions, providing an execution framework for functions e.g. C++ code, **set** of **SQL statements** to be maintained...

...Exposure of **metadata** regarding a functions' input and output parameters allows an engine to track input/output **relationship** between the functions and define order of execution...

...The drawing shows a block diagram illustrating the **relationship** between functions being tracked, **metadata objects**, execution **objects** and **manager objects** according to

an embodiment of the invention...

...**Metadata objects** (120...

...Execution **objects** (125...

...**Manager object** (130
International Patent Class (Main): **G06F-017/30**
Manual Codes (EPI/S-X): **T01-F05A**...

...**T01-J20B**...

...**T01-S03**

US006338069B1

(12) **United States Patent**
Ghatate

(10) **Patent No.: US 6,338,069 B1**
(45) **Date of Patent: Jan. 8, 2002**

(54) **METHOD AND APPARATUS FOR MANAGING FUNCTIONS**

(75) Inventor: **Bhalchandra Ghatate**, Austin, TX (US)

(73) Assignee: **Daman, Inc.**, Austin, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/210,340**

(22) Filed: **Dec. 11, 1998**

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** **707/103**
(58) **Field of Search** 707/103

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,481,718	A *	1/1996	Ryu et al.	709/316
5,497,491	A *	3/1996	Mitchell et al.	395/700
5,668,958	A *	9/1997	Bendert et al.	710/128
5,717,925	A *	2/1998	Harper et al.	707/102
5,721,911	A *	2/1998	Ha et al.	707/100
5,761,678	A *	6/1998	Bendert et al.	707/204
5,991,768	A *	11/1999	Sun et al.	707/104
6,128,621	A *	10/2000	Weisz	707/103
6,134,559	A *	10/2000	Brumme et al.	707/103
6,195,662	B1 *	2/2001	Ellis et al.	707/103
6,226,628	B1 *	5/2001	Forbes	707/1
6,226,788	B1 *	5/2001	Schoening et al.	717/6
6,253,239	B1 *	6/2001	Shklar et al.	709/217
6,263,313	B1 *	7/2001	Milsted et al.	705/1

OTHER PUBLICATIONS

Integration Solutions for the Real-Time Enterprise: EIT-Enterprise Integration Template, Template Software (Table of Contents: 1 pg.; Contents: 3–28; Appendix I: 29–30; Appendix II: 31–32).

Object Technology, A Manager's Guide, Second Edition by David A. Taylor, Ph.D., 2/98 (Contents:v–vii; Acknowledgements: ix–x; Introduction: xi–xii; How to Read the Book; xiii–xvi; Chapter 1–9; 1–158; Appendix: 159–175; Suggested Reading: 177–182; Glossary: 183–198; Index: 199–205.

Upstream, A White Paper, Business Process Engines, A New Catagory of Server Software, Will Burst the Barriers in Distributed Application Performance by John Rymer, Upstream Consulting (pp. 1–16; Appendix: 16–19).

ADT Mar. 1997—Viewed as an effective new way to identify core business requirements, business rule thinking A/D is taking shape, Software Engineering (http://ww.adt-mag.com/pub/mar97/softeng. (15 pgs.).

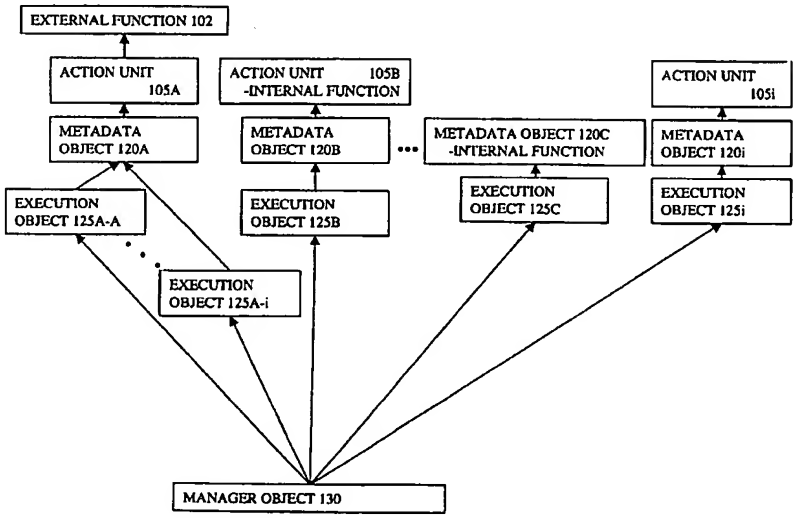
* cited by examiner

Primary Examiner—Wayne Amsbury
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman, LLP

(57) **ABSTRACT**

A method and apparatus for managing functions (e.g., that express business rules) to allow calling functions, maintaining functions, and providing of an execution framework for functions. In one embodiment, there are a number of functions to be maintained. An object technology infrastructure is formed to store data and metadata for the functions. For example, metadata about a function can include data describing what that function does, a "cost" associated with that function, how to execute that function, the input and output parameters required by that function. The exposure of the metadata regarding the functions' input and output parameters allows an engine to track input/output relationships between the functions and, in essence, define the order of execution.

46 Claims, 21 Drawing Sheets



execute methods of the execution objects in the level objects of the EXECUTION_PATH_COLLECTION structure are applied in level order.

It should be noted that application of each EXECUTE method requires the operations illustrated in FIG. **19** to be performed. It should be understood that application of one or more of these EXECUTE methods could also result in a determination that the values for one or more required inputs of an underlying function is missing. In other words, this operation can be recursive in nature.

According to the embodiment shown in FIG. **20**, for each key in the NEED_COLLECTION (see block **2015**) it is verified that a value was acquired. In particular, block **2020** determines if a value was returned. If a value was not returned, control passes to block **2030** in which a flag is set for debugging purposes. However, if a value was returned, then the SET_PARAM method is applied with the returned value to associate the value with the EXECUTION object. While one embodiment is described in which verification that a value was returned for each key in the need collection is performed, alternative embodiments can avoid this verification and/or provide for this verification in other areas.

In this manner, values for any of the missing input parameters to the underlying function are located as part of applying the EXECUTE method from the EXECUTION object.

Relationship Objects

In one embodiment, the integration layer also includes two types of objects, base objects and relationship objects. The base objects describe the disparate integration sources A-i. The relationship objects describe relationships between the base objects, as well as relationships between the relationship objects themselves.

Specifically, the common concept of encapsulation in object technology is to place data with its' associated methods together in a single object. In contrast, the integration layer is developed such that those methods that express a relationship (referred to herein as "relationship methods") are placed in separate objects—the relationship objects. In this manner, a higher degree of encapsulation is provided. While relationship methods within relationship objects are similar to methods commonly found in object technology in that they can be applied to the objects that contain them, relationship methods within relationship objects are different in that they often are applied to other objects, including base objects and other relationship objects. For a further description of relationship objects, see "A Method and Apparatus for Providing Relationship Objects and Various Features to Relationship and Other Objects," filed Sep. 30, 1998, by Bhalchandra Ghatate, which is herein incorporated by reference.

As previously indicated, in one embodiment of the invention, relationship object(s) are used to provide for one or more SATISFY_RELATIONSHIP routines. In this manner, different types of satisfy relationship routines can be provided and selected from using the relationship object techniques.

Alternative Embodiments

While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described. The method and apparatus of the invention can be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of limiting on the invention.

What is claimed is:

1. A machine readable medium having stored thereon:

a function requiring a set of one or more input parameters;

a first object including,

a first structure storing a key for each of said set of input parameters, and

an action method, which when applied by a processor, causes that processor to invoke an action unit; and

a second object including,

a first structure to store data for identifying, for each of said set of input parameters, the corresponding key and a value for that input parameter,

a second structure identifying said first object, and

an execute method, which when applied by a processor, causes that processor to apply said action method;

said action unit including instructions, which when executed by a processor, cause that processor to,

access said values, and

invoke said function using said values as input parameters.

2. The machine readable medium of claim **1**, wherein said function is contained within said action unit.

3. The machine readable medium of claim **1**, wherein said first structure in said second object has stored therein, for at least one of said set of input parameters, the corresponding key and a value for that input parameter.

4. The machine readable medium of claim **1**, wherein:

the data in said first structure of said second object identifies a third object; and

said third object includes a structure to store, for one or more of said set of input parameters, the corresponding key and value for that input parameter.

5. The machine readable medium of claim **1**, wherein:

the data in said first structure of said second object identifies a third object; and

said third object includes,

a first structure to store a plurality of context objects, each of said plurality of context objects to store, for one or more of said set of input parameters, the corresponding key and a value for that input parameter, and

a second structure to store data identifying one of said plurality of context objects as a default context object.

6. The machine readable medium of claim **1**, wherein:

said function provides a set of one or more output parameters;

said first structure in said first object also storing a key for each of said set of output parameters.

7. A machine readable medium having stored thereon:

a function requiring a set of one or more input parameters;

a first object including,

a first structure storing a key for each of said set of input parameters, and

an action method, which when applied by a processor, causes that processor to invoke said function; and

a second object including,

a first structure identifying a third object,

a second structure identifying said first object, and

an execute method, which when applied by a processor, causes that processor to apply said action method; and

said third object including,

a first structure to store a plurality of context objects, each of said plurality of context objects to store, for

one or more of said set of input parameters, the corresponding key and a value for that input parameter, and

a second structure to store data identifying one of said plurality of context objects as a default context object.

8. The machine readable medium of claim 7 further having stored thereon:

an action unit including said function as a method.

9. The machine readable medium of claim 7 further having stored thereon:

an action unit including instructions, which when executed by a processor, cause that processor to invoke said function; and

wherein said action method, when applied, causes the execution of the instructions in said action unit.

10. The machine readable medium of claim 7 further having stored thereon:

an action unit including instructions, which when executed by a processor, cause that processor to,

access values for one or more of said set of input parameters from a selected one of said plurality of context objects that is passed to said action unit, and

invoke said function using said values as input parameters; and

wherein said action method, when applied, causes said function to be invoked through the instructions in said action unit.

11. The machine readable medium of claim 10, wherein:

said second object also includes a first structure storing data identifying, for one or more of said set of input parameters, the corresponding key and a value for that input parameter; and

said instructions of said action unit, when executed by a processor, also cause that processor to access values for one or more of said set of input parameters from said first structure.

12. The machine readable medium of claim 11, wherein:

said instructions of said action unit, when executed by a processor, also cause that processor to access values for one or more of said set of input parameters from said default context object.

13. The machine readable medium of claim 11, wherein:

said instructions of said action unit, when executed by a processor, also cause that processor to access values for one or more of said set of input parameters from said default context object.

14. The machine readable medium of claim 7 further having stored thereon:

said second object also includes a first structure to store data identifying, for one or more of said set of input parameters, the corresponding key and a value for that input parameter; and

an action unit including instructions, which when executed by a processor, cause that processor to,

access values for said set of input parameters from said first structure of said second object, a selected one of said plurality of context objects that is passed to said action unit, and said default context object, and

invoke said function using said values as input parameters.

15. A machine readable medium having stored thereon:

a plurality of functions for one or more applications, each of said plurality of functions requiring one or more input parameters, the input parameters required by said plurality of functions collectively defining a set of parameter kinds irrespective of data type, each parameter kind in said set being assigned a unique key;

a metadata object corresponding to each of said plurality of functions, each said metadata object storing data to locate the corresponding one of said plurality of functions, each said metadata object also storing the unique key for each input parameter required by the corresponding one of said plurality of functions; and

each metadata object having one or more corresponding execution objects, each execution object including a structure storing data to identify a value for each input parameter of the one of said plurality of functions identified by the corresponding metadata object.

16. The machine readable medium of claim 15, wherein each parameter of said plurality of functions is of one of a plurality of data types each supporting a range of values, wherein different data is categorized irrespective of data type, and wherein each category of data defines one of the parameter kinds.

17. The machine readable medium of claim 15 further having stored thereon:

an action unit for each of said plurality of functions, each of said action units including instructions, which when executed by a processor, cause that processor to invoke the corresponding one of said plurality of functions; and

wherein the data in each of said metadata objects for locating the corresponding one of said plurality of functions identifies the corresponding one of said action units; and

wherein each of said metadata objects includes an action method, which when applied, causes the execution of the instructions in the corresponding one of said action units.

18. The machine readable medium of claim 17, wherein:

each execution object includes a method, which when applied, causes said action method of the corresponding metadata object to be applied for that business rule;

each action method, when applied responsive to the method of an execution object, causes the instructions in the corresponding action unit to be executed for that business rule; and

the instructions in each action unit, when applied responsive to an action method responsive to the method of an execution object, causes the values of that business rule to be accessed and said function of that action unit to be invoked with said values as input parameters.

19. The machine readable medium of claim 15, wherein at least one of said execution objects stores the key and a corresponding value for at least one input parameter of the corresponding one of said plurality of functions.

20. The machine readable medium of claim 15, wherein:

at least one of said execution objects includes a structure identifying a manager object; and

said manager object includes a structure to store the key and a corresponding value for at least on input parameter of the one of said plurality of functions corresponding to the at least one of said execution objects.

21. The machine readable medium of claim 15, wherein:

at least one of said execution objects includes a structure identifying a manager object; and

said manager object includes,

a first structure to store a plurality of context objects, each of said plurality of context objects to store the

key and a corresponding value for one or more input parameters to at least certain of the plurality of functions, and

a second structure to store data identifying one of said plurality of context objects as a default context object.

22. The machine readable medium of claim 15, wherein:

each of said plurality of functions provides one or more output parameters, the input and output parameters of said plurality of functions collectively defining said set of parameter kinds irrespective of data type;

each said metadata object also storing the unique key for each input and output parameter of the corresponding one of said plurality of functions.

23. A machine readable medium having stored thereon sequences of instructions, which when executed by a set of one or more processors, cause said set of one or more processors to perform the acts of:

applying a first method from a first execution object, said first execution object identifying a first metadata object corresponding to a first function, said first function requiring one or more input parameters, said first metadata object storing data describing each input parameter of said first function, said first method causing the acts of,

accessing the data describing each input parameter of said first function from said first metadata object;

attempting to match values associated with said first execution object to each input parameter of said first function as described by the data; and

determining a value is missing for at least a first input parameter to said first function.

24. The machine readable medium of claim 23, wherein said first method further causes the acts of:

locating a second metadata object corresponding to a second function having one or more output parameters, said second metadata object storing data describing each output parameter of said second function;

determining the missing first input parameter is an output parameter of said second function; and

executing said second function to acquire the missing value.

25. The machine readable medium of claim 24, wherein said first method further causes the act of:

associating the acquired value with said first execution object; and

executing said first function using the acquired value now associated with said first execution object as the first input parameter.

26. The machine readable medium of claim 23, wherein said first method further causes the acts of:

determining a set of metadata objects that each have stored therein data describing an output parameter that matches one or more of the missing input parameters, wherein each metadata object in said set corresponds to a different function having a set of one or more output parameters, each metadata object in said set storing data describing said set of output parameters for the corresponding function; and

executing the functions corresponding to the set of metadata objects to acquire said set of missing values;

associating the acquired values with the first execution object; and executing said first function using the values associated with the first execution object as input parameters.

27. The machine readable medium of claim 23, wherein said attempting further includes:

accessing a structure in said first execution object, said structure in said first execution object to store values for one or more said set of input parameters to said first function.

28. The machine readable medium of claim 23, wherein said attempting further includes:

accessing a structure in a manager object identified by a structure in said first execution object, said structure in said manager object identifying a default one of a plurality of context objects, each of said plurality of context objects to store values for one or more of the input parameters to said first function; and

accessing said values from said default context object.

29. The machine readable medium of claim 23, wherein said sequences of instructions, when executed, cause said set of processors to further perform the acts of:

applying a first method from a second execution object, said second execution object also identifying said first metadata object, said first method of said second execution object causing the acts of,

accessing the data describing each input parameter of said first function from said first metadata object;

attempting to match values associated with said second execution object to each input parameter of said first function as described by the data in said first metadata object; and

determining one or more values are missing for at least certain input parameters of said first function.

30. A machine readable medium having stored thereon sequences of instructions, which when executed by a set of one or more processors, cause said set of one or more processors to perform the acts of:

applying a first method from a first execution object, said first execution object identifying a first of a plurality of metadata objects, each of said plurality of metadata objects identifying a different function, each of said functions having input and output parameters, wherein one or more parameters for different functions are the same, the parameters for the different functions collectively defining a set of parameter kinds, each parameter kind in said set of parameter kinds being assigned a unique key, each of said plurality of metadata objects storing the unique keys assigned the input and output parameters of the function they identify, said first method causing the acts of,

accessing the key for each input parameter stored in said first metadata object;

attempting to match parameter values associated with said first execution object to each of the accessed keys; and

determining parameter values are missing for a set including at least one of the accessed keys.

31. The machine readable medium of claim 30, wherein each parameter of said functions is of one of a plurality of data types each supporting a range of values, wherein different data is categorized irrespective of data type, and wherein each category of data defines one of the set of parameter kinds.

32. The machine readable medium of claim 30, wherein said first method further causes the acts of:

locating a set of one or more of said plurality of metadata objects that collectively store each of the set of keys; and

executing the functions corresponding to the set of metadata objects to acquire said set of missing parameter values as outputs of the functions;

associating the acquired parameter values with the first execution object; and

executing the function identified by the first metadata object using the parameter values associated with the first execution object as input parameters.

33. The machine readable medium of claim **30**, wherein said attempting further includes:

accessing a structure in said first execution object, said structure in said first execution object to store values for one or more said set of input parameters to the function identified by the first metadata object.

34. The maching readable medium of claim **30**, wherein said attempting further includes:

accessing a structure in a manager object identified by a structure in said first execution object, said structure in said manager object identifying a default one of a plurality of context objects, each of said plurality of context objects to store values for one or more of the input parameters to said first function; and

accessing one or more of said values from said default context object.

35. The machine readable medium of claim **30**, wherein said sequences of instructions, when executed, cause said set of processors to further perform the acts of:

applying a first method from a second execution object, said second execution object also identifying said first metadata object, said first method of said second execution object causing the acts of,

accessing the key for each input parameter stored in said first metadata object;

attempting to match values associated with said second execution object to each of the accessed keys; and

determining parameter values are missing for a set including at least one of the accessed keys.

36. The machine readable medium of claim **30**, wherein said attempting further includes:

accessing from a first structure in said first execution object a value for a first input parameter to said function identified by said first metadata object;

accessing, from a first of a set of context objects that was passed, a value for a second input parameter to said first function, said set of context objects being stored in a first structure of a manager object, each of said set of context objects to store values for one or more of the input parameters to said first function, said manager object identifying one of said set of context objects as a default context object; and

accessing from said default context object a value for a third input parameter to said first function.

37. A machine readable medium having stored thereon sequences of instructions, which when executed by a set of one or more processors, cause said set of one or more processors to perform the acts of:

receiving a request to locate a function that provides a particular parameter kind as an output;

locating a metadata object having stored therein data identifying said particular parameter kind, said metadata object identifying a function and storing said data to indicate the particular parameter kind is an output parameter of said function; and

providing an execution object that identifies said metadata object, wherein said execution object includes,

structure to identify values for a set of one or more input parameters to said function, and

a method, which when applied, causes said function to be invoked using the values identified by said structure as input parameters.

38. The machine readable medium of claim **37**, wherein:

said function has a plurality of parameters, said metadata object stores data identifying each kind of said plurality of parameters.

39. The machine readable medium of claim **38**, wherein each parameter of said function is of one of a plurality of data types each supporting a range of values, wherein different data is categorized irrespective of data type, and wherein each category of data defines one of the parameter kinds.

40. The machine readable medium of claim **37**, wherein:

said metadata object includes an action method, which when applied by a processor, causes said processor to invoke said function; and

said method in said execution object, when applied, causes said action method to be applied.

41. A machine readable medium having stored thereon sequences of instructions, which when executed by a set of one or more processors, cause said set of one or more processors to perform the acts of:

receiving a request to locate a function that provides a particular output parameter, wherein each of a plurality of metadata objects identify a different function having one or more output parameters, said output parameters for the different functions collectively defining a set of parameter kinds, each parameter kind in said set being assigned a unique key, each of said plurality of metadata objects storing the unique keys assigned the output parameters of the function they identify;

locating a first of said plurality of metadata objects that stores the unique key for the particular output parameter; and

creating an execution object that identifies said first metadata object, wherein said execution object includes,

a structure to identify values for a set of one or more input parameters to said function identified by said first metadata object, and

a method, which when applied, causes said function identified by said first metadata object to be invoked using the values identified by said structure as input parameters.

42. The machine readable medium of claim **41**, wherein each parameter of said functions is of one of a plurality of data types each supporting a range of values, wherein different data is categorized irrespective of data type, and wherein each category of data defines one of the parameter kinds.

43. The machine readable medium of claim **41**, wherein said sequences of instructions, when executed, cause said set of processors to further perform the acts of:

applying said method from said execution object, wherein both said input and output parameters for the different functions collectively define said set of parameter kinds, each of said plurality of metadata objects storing the unique keys assigned the input and output parameters of the function they identify, said method from said execution object causing the acts of,

accessing the key for each input parameter stored in said first metadata object;

attempting to match parameter values associated with said execution object to each of the accessed keys; and

executing the function identified by said first metadata object using the parameter values associated with the execution object as input parameters.

44. The machine readable medium of claim 41, wherein said attempting further causes the acts of:

determining parameter values are missing for a set including at least one of the accessed keys.

locating a set of one or more of said plurality of metadata objects that collectively store each of the set of keys for output parameters; and

executing the functions corresponding to the plurality of metadata objects to acquire said set of missing parameter values as outputs of the functions; and

associating the acquired parameter values with first execution object.

45. A machine readable medium having stored thereon sequences of instructions, which when executed by a set of one or more processors, cause said set of one or more processors to perform the acts of:

applying a first method from a first execution object, said first execution object identifying a first of a plurality of metadata objects, each of said plurality of metadata objects identifying a different function, each of said functions having input and output parameters, wherein one or more parameters for different functions are the same, said parameters for the different functions collectively defining a set of parameter kinds, each parameter kind in said set of parameter kinds being assigned a unique key, each of said plurality of metadata objects storing the unique keys assigned the input and output parameters of the function they identify, said first method causing the acts of,

accessing the key for each input parameter stored in said first metadata object;

associating with said first execution object a value stored as part of a first of a set of context objects that was passed, said value stored for use as a first input parameter, said set of context objects being stored in a first structure of a manager object, each of said set of context objects to store values for one or more of the input parameters to said first function; and

executing said first function using the parameter values associated with the first execution object as input parameters.

46. The machine readable medium of claim 45, wherein said first method further causes the acts of:

associating with said first execution object a value stored as part of one of said set of context objects identified by said manager object as a default context object, said value stored for a second input parameter to said first function.

* * * * *

46/3,K/37 (Item 37 from file: 350) **Links**
Derwent WPIX

015684733 **Image available**
WPI Acc No: 2003-746922/200370
Related WPI Acc No: 2005-272127; 2005-272128
XRPX Acc No: N03-598603
Schema generation method in computer system, involves generating schema to represent hierarchies of inter-object relationships between objects in data polyarchy, based on attribute values of objects
Patent Assignee: BOOTH J H (BOOT-I); CAMERON K (CAME-I); CLEMENT L (CLEM-I); MACLEOD S P (MACL-I); ROBERTSON G G (ROBE-I); MICROSOFT CORP (MICT)
Inventor: BOOTH J H; CAMERON K; CLEMENT L; MACLEOD S P; ROBERTSON G G
Number of Countries: 001 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 20030177139	A1	20030918	US 2001995415	A	20011126	200370	B
US 6944626	B2	20050913	US 2001995415	A	20011126	200560	

Priority Applications (No Type Date): US 2001995415 A 20011126
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20030177139	A1		35	G06F-007/00	
US 6944626	B2			G06F-017/00	

Schema generation method in computer system, involves generating schema to represent hierarchies of inter-object relationships between objects in data polyarchy, based on attribute values of objects

Abstract (Basic):
... A schema is generated to represent multiple hierarchies of inter-**object relationships** between various **objects** in data polyarchy, based on attribute values of the **objects**.
... 3) polyarchical **query language** data structure...

...including personal computer (claimed), server computer, laptop computer, handheld computer, multiprocessor system, microprocessor based system, **set**-top box, programmable consumer electronics, minicomputer and mainframe computer connected to network such as Internet...

...Since the data polyarchy complex inter-**object relationships** are dynamically generated, the manual intervention from a network administrator to **configure** inter-**object relationship**, is not required...
...**optional** data servers (106...

...**client** computer (110...

...computer **executable** instructions (116...

...polyarchical data **set** (122
...Title Terms: **OBJECT;**
International Patent Class (Main): **G06F-007/00**...

...**G06F-017/00**
Manual Codes (EPI/S-X): **T01-F05E**...

...**T01-F07**...

...**T01-N02B1A**...

...**T01-S03**

US006944626B2

(12) **United States Patent**
Cameron et al.

(10) **Patent No.: US 6,944,626 B2**
(45) **Date of Patent: Sep. 13, 2005**

(54) **DYNAMICALLY GENERATED SCHEMA REPRESENTING MULTIPLE HIERARCHIES OF INTER-OBJECT RELATIONSHIPS**

(75) Inventors: **Kim Cameron**, Bellevue, WA (US); **Stewart P. MacLeod**, Woodinville, WA (US); **George G. Robertson**, Seattle, WA (US); **James H. Booth**, Issaquah, WA (US); **Luc Clement**, Sammamish, WA (US)

(73) Assignee: **Microsoft Corp.**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 561 days.

(21) Appl. No.: **09/995,415**

(22) Filed: **Nov. 26, 2001**

(65) **Prior Publication Data**

US 2003/0177139 A1 Sep. 18, 2003

(51) **Int. Cl.**[7] **G06F 17/00**
(52) **U.S. Cl.** ..... **707/103 R**; 707/100; 707/102
(58) **Field of Search** ..... 707/100, 102, 707/103 R, 103 Y

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,802,511	A		9/1998	Kouchi et al.	
5,859,978	A		1/1999	Sonderegger et al.	
6,223,145	B1	*	4/2001	Hearst	703/22
6,285,366	B1	*	9/2001	Ng et al.	345/853
6,434,564	B2	*	8/2002	Ebert	707/100
6,442,557	B1	*	8/2002	Buteau et al.	707/102
6,463,420	B1		10/2002	Guidice et al.	
6,564,263	B1	*	5/2003	Bergman et al.	709/231
6,643,652	B2		11/2003	Helgeson et al.	
6,708,161	B2		3/2004	Tenorio et al.	
2001/0047385	A1		11/2001	Tuatini	
2002/0169744	A1	*	11/2002	Cooke et al.	
2004/0002982	A1		1/2004	Ersek et al.	

OTHER PUBLICATIONS

Chita, Christian (Hierarchy Visualization).*
T. Howes, et al., "The LDAP URL Format", The Internet Society, Dec. 1997, 8 pages.
M. Wahl, "A Summary of the X.500(96) User Schema for use with LDAPv3", The Internet Society, Dec. 1997, 20 pages.

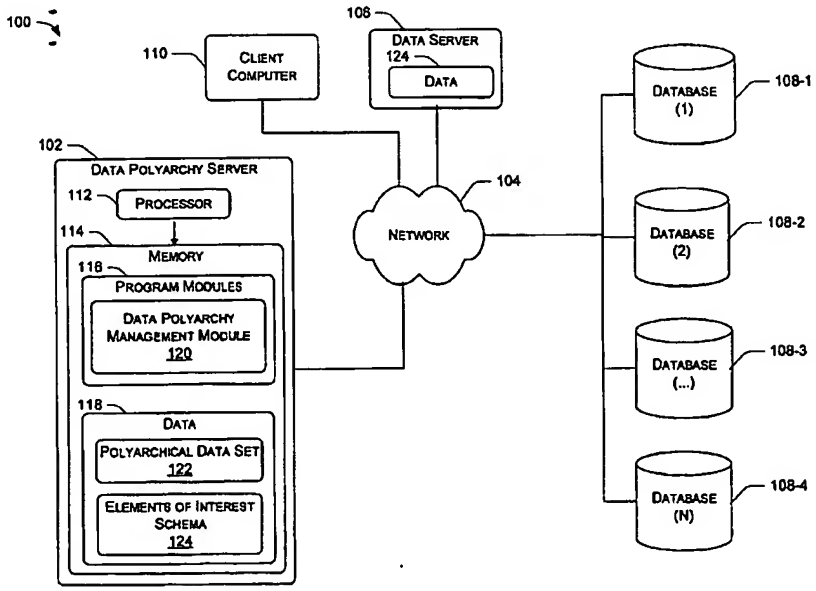
(Continued)

Primary Examiner—Frantz Coby
Assistant Examiner—Haythim Alaubaidi
(74) *Attorney, Agent, or Firm*—Lee & Hayes PLLC

(57) **ABSTRACT**

The described arrangements and procedures provide for interfacing (e.g., managing, presenting, etc.) with complex and often elastic inter-object relationships between objects in a data polyarchy. Specifically, a schema is dynamically generated by a server to represent multiple hierarchies of inter-object relationships between objects in a data polyarchy. The schema indicates or lists each attribute or element of interest in the data polyarchy. The schema further indicates any of one or more dimensions of inter-object relationships within which objects that comprise at least a subset of the listed the elements of interest participate. Thus, the schema indicates how to interface with the data polyarchy, which represents multiple hierarchies of inter-object relationships based on the values of attributes of the represented objects.

70 Claims, 14 Drawing Sheets



US 20030177139A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0177139 A1

Cameron et al. (43) Pub. Date: Sep. 18, 2003

(54) DYNAMICALLY GENERATED SCHEMA REPRESENTING MULTIPLE HIERARCHIES OF INTER-OBJECT RELATIONSHIPS

(76) Inventors: Kim Cameron, Bellevue, WA (US); Stewart P. MacLeod, Woodinville, WA (US); George G. Robertson, Seattle, WA (US); James H. Booth, Issaquah, WA (US); Luc Clement, Sammamish, WA (US)

Correspondence Address:
LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

(21) Appl. No.: 09/995,415

(22) Filed: Nov. 26, 2001

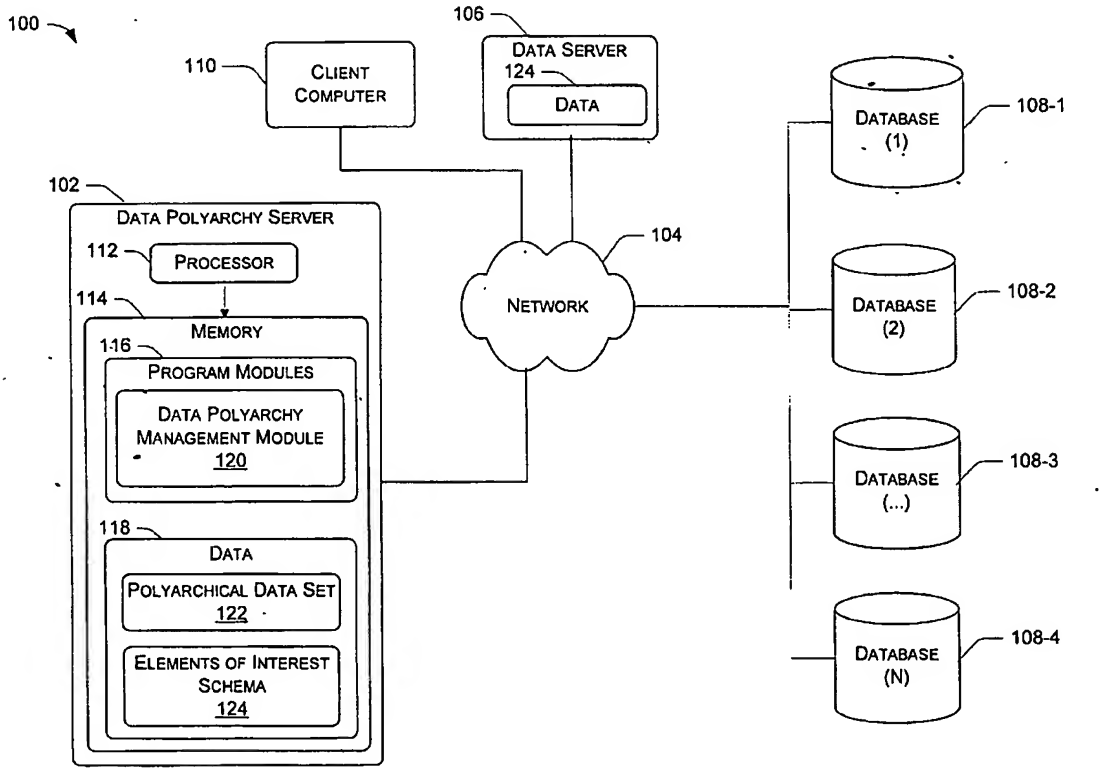
Publication Classification

(51) Int. Cl.[7] G06F 7/00

(52) U.S. Cl. ..... 707/103 R

(57) ABSTRACT

The described arrangements and procedures provide for interfacing (e.g., managing, presenting, etc.) with complex and often elastic inter-object relationships between objects in a data polyarchy. Specifically, a schema is dynamically generated by a server to represent multiple hierarchies of inter-object relationships between objects in a data polyarchy. The schema indicates or lists each attribute or element of interest in the data polyarchy. The schema further indicates any of one or more dimensions of inter-object relationships within which objects that comprise at least a subset of the listed the elements of interest participate. Thus, the schema indicates how to interface with the data polyarchy, which represents multiple hierarchies of inter-object relationships based on the values of attributes of the represented objects.



objects. In this manner, complex real-world objects are represented with respect to the particular objects themselves, with respect to any set of decomposed sub-entities, or sub-objects that are related to the particular objects. These inter-object relationships are managed and navigated using a data polyarchy schema **124** that has been generated to access elements of interest in the data polyarchy **122**.

[0155] Although the described subject matter to generate and manage polyarchies of data relationships has been described in language specific to structural features and/or methodological operations, it is to be understood that the subject defined in the appended claims is not necessarily limited to the specific features or operations described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed present invention.

1. In a computer system, a method comprising dynamically generating a schema to represent multiple hierarchies of inter-object relationships between a plurality of objects in a data polyarchy, the schema being generated based on values of attributes of the objects.

2. A method as recited in claim 1, wherein the inter-object relationships comprise a flat relationship, a hierarchical relationship, and multiple intersecting hierarchies of relationships.

3. A method as recited in claim 1, where the objects comprise enterprise resource planning (ERP) objects, directory based objects, or database objects.

4. A method as recited in claim 1, wherein generating the schema further comprises:

identifying a plurality of attributes of interest based on values of attributes of the objects;

identifying one or more dimensions of inter-object relationships within which objects that comprise at least a subset of the attributes of interest participate.

5. A method as recited in claim 1, wherein the schema is designed to provide access control to organizational resources.

6. A method as recited in claim 1, further comprising communicating the schema to a client, the schema identifying how the client can access objects in the data polyarchy.

7. A method as recited in claim 1, further comprising:

receiving a request from a client based on the schema; and

responsive to receiving the request:

accessing an object of the objects based on the request;

transforming the object into transformed data that expresses any inter-object relationship between the object and any other object of the objects based on the request; and

issuing the transformed data to the client.

8. A method as recited in claim 7, wherein the transformed data expresses the inter-object relationships with respect to other objects in a same dimension or other objects in a different dimension, the same and/or the different dimension being indicated by the request.

9. A method as recited in claim 7, wherein the request comprises a limiting attribute to limit the transformed data by presenting the one or more objects only with respect to the limiting attribute.

10. A method as recited in claim 7, wherein the request queries for information corresponding to an object in the data polyarchy with respect to one or more particular dimensions.

11. A method as recited in claim 7, wherein the request comprises a dimension indicator to specify one or more hierarchies within which the data is to be presented in the transformed data.

12. A method as recited in claim 7, wherein the request further comprises a distinguishing attribute, a classifying attribute, or a locating attribute.

13. A method as recited in claim 7, wherein the request comprises a dimension information modifier to specify a particular direction and a particular depth to retrieve information from the data polyarchy.

14. A method as recited in claim 13, wherein the dimension information modifier is a siblings indication to retrieve all objects with a same parent as a current object in the data polyarchy.

15. A method as recited in claim 7, wherein the request indicates that at least one subset of the objects comprise a similar attribute; and wherein accessing one or more objects further comprises:

retrieving the one or more objects in a manner that is independent of any hierarchical data relationship between the data objects in the at least one subset.

16. A method as recited in claim 15, wherein the similar attribute comprises a logical domain selected from a distinguishing domain, a locating domain or a classifying domain.

17. A method as recited in claim 7, wherein the request corresponds to at least a first and second subset of the objects, the request comprising a logical modifier that specifies an operation, and wherein the method further comprises:

responsive to receiving the request, identifying at least a portion of the first and second subsets of directory objects in the polyarchical data set; and

wherein transforming the one or more objects further comprises performing the operation on the first and second subsets.

18. A method as recited in claim 17, wherein the logical modifier is a Boolean modifier.

19. A method as recited in claim 17, wherein the operation comprises any combination of filtering, union, intersection, join, and/or exclusion operations.

20. A method as recited in claim 7, wherein accessing the object further comprises accessing the object in a manner that is independent of any inter-object relationship between the object and any other object of the objects in a manner that is independent of any definition of a hierarchy in the data polyarchy.

21. A method as recited in claim 7, wherein accessing the object further comprises querying the data polyarchy for the object.

22. A method as recited in claim 7, wherein accessing the object further comprises managing, manipulating, or modifying the object or a relationship between the object and one of more of the other objects.

23. A computer-readable medium having computer-executable instructions comprising instructions for:

dynamically generating a schema to represent multiple hierarchies of inter-object relationships between a plurality of objects in a data polyarchy, the schema being

generated based on values of attributes of the objects, the schema indicating each attribute of interest in the data polyarchy, the schema further indicating any of one or more dimensions of inter-object relationships within which objects that comprise at least a subset of the attributes of interest participate.

24. A computer-readable medium as recited in claim 23, wherein the inter-object relationships comprise a flat relationship, a hierarchical relationship, and multiple intersecting hierarchies of relationships.

25. A computer-readable medium as recited in claim 23, where the objects comprise enterprise resource planning (ERP) objects, directory based objects, or database objects.

26. A computer-readable medium as recited in claim 23, further comprising computer-executable instructions for communicating the schema to a client to indicate how the client is to interface with the objects in the data polyarchy.

27. A computer-readable medium as recited in claim 23, further comprising computer-executable instructions for:

receiving a request from a client based on the schema;

responsive to receiving the request:

accessing at least one object in the data polyarchy based on the request;

transforming the at least one object into transformed data that expresses any inter-object relationship between the at least one object and any other objects of the objects based on the request; and

issuing the transformed data to the client.

28. A computer-readable medium as recited in claim 27, wherein the transformed data expresses the inter-object relationships with respect to other objects in a same dimension or other objects in a different dimension, the same and/or the different dimension being indicated by the request.

29. A computer-readable medium as recited in claim 27, wherein the request comprises a limiting attribute to limit the transformed data by presenting the one or more objects only with respect to the limiting attribute.

30. A computer-readable medium as recited in claim 27, wherein the request queries for information corresponding to an object in the data polyarchy with respect to one or more particular dimensions.

31. A computer-readable medium as recited in claim 27, wherein the request comprises a dimension indicator to specify one or more hierarchies within which the data is to be presented in the transformed data.

32. A computer-readable medium as recited in claim 27, wherein the request further comprises a distinguishing attribute, a classifying attribute, or a locating attribute.

33. A computer-readable medium as recited in claim 27, wherein the request comprises a dimension information modifier to specify a particular direction and a particular depth to retrieve information from the data polyarchy.

34. A computer-readable medium as recited in claim 33, wherein the dimension information modifier is a siblings indication to retrieve all objects with a same parent as a current object in the data polyarchy.

35. A computer-readable medium as recited in claim 27 wherein the request indicates that at least one subset of the objects comprise a similar attribute; and wherein the computer-executable instructions for accessing one or more objects further comprise instructions for:

retrieving the one or more objects in a manner that is independent of any hierarchical data relationship between the data objects in the at least one subset.

36. A computer-readable medium as recited in claim 35, wherein the similar attribute comprises a logical domain selected from a distinguishing domain, a locating domain or a classifying domain.

37. A computer-readable medium as recited in claim 27, wherein the request corresponds to at least a first and second subset of the objects, the request comprising a logical modifier that specifies an operation, and wherein the computer-executable instructions further comprise instructions for:

responsive to receiving the request, identifying at least a portion of the first and second subsets of directory objects in the polyarchical data set; and

wherein transforming the one or more objects further comprises performing the operation on the first and second subsets.

38. A computer-readable medium as recited in claim 37, wherein the logical modifier is a Boolean modifier.

39. A computer-readable medium as recited in claim 37, wherein the operation comprises any combination of filtering, union, intersection, join, and/or exclusion operations.

40. A computer-readable medium as recited in claim 27, wherein accessing the at least one object further comprises accessing the at least one object in a manner that is independent of any inter-object relationship between the at least one object and any other object of the objects in a manner that is independent of any definition of a hierarchy in the data polyarchy.

41. A computer-readable medium as recited in claim 27, wherein accessing the at least one object further comprises querying the data polyarchy for the at least one object.

42. A computer-readable medium as recited in claim 27, wherein accessing the at least one object further comprises managing, manipulating, or modifying the at least one object or a relationship between the at least one object and one of more different objects of the objects.

43. A computer comprising:

a memory comprising the computer-executable instructions; and

a processor coupled to the memory, the processor being configured to fetch and execute the computer-executable instructions for:

dynamically generating a schema to represent multiple hierarchies of inter-object relationships between a plurality of objects in a data polyarchy, the schema being generated based on values of attributes of the objects, the schema indicating each attribute of interest in the data polyarchy, the schema further indicating any of one or more dimensions of inter-object relationships within which objects that comprise at least a subset of the attributes of interest participate.

44. A computer as recited in claim 43, wherein the inter-object relationships comprise a flat relationship, a hierarchical relationship, and multiple intersecting hierarchies of relationships.

45. A computer as recited in claim 43, where the objects comprise enterprise resource planning (ERP) objects, directory based objects, or database objects.

46. A computer as recited in claim 43, wherein the computer-executable instructions further comprise instructions for communicating the schema to a client to indicate how the client is to interface with the objects in the data polyarchy.

47. A computer as recited in claim 43, wherein the computer-executable instructions further comprise instructions for:

receiving a request from a client based on the schema;

responsive to receiving the request:

accessing one or more objects in the data polyarchy based on the request;

transforming the one or more of the objects into transformed data that expresses any inter-object relationships based on the request; and

issuing the transformed data to the client.

48. A computer as recited in claim 47, wherein the transformed data expresses the inter-object relationships with respect to other objects in a same dimension or other objects in a different dimension, the same and/or the different dimension being indicated by the request.

49. A computer as recited in claim 47, wherein the request comprises a limiting attribute to limit the transformed data by presenting the one or more objects only with respect to the limiting attribute.

50. A computer as recited in claim 47, wherein the request queries for information corresponding to an object in the data polyarchy with respect to one or more particular dimensions.

51. A computer as recited in claim 47, wherein the request comprises a dimension indicator to specify one or more hierarchies within which the data is to be presented in the transformed data.

52. A computer as recited in claim 47, wherein the request further comprises a distinguishing attribute, a classifying attribute, or a locating attribute.

53. A computer as recited in claim 47, wherein the request comprises a dimension information modifier to specify a particular direction and a particular depth to retrieve information from the data polyarchy.

54. A computer as recited in claim 53, wherein the dimension information modifier is a siblings indication to retrieve all objects with a same parent as a current object in the data polyarchy.

55. A computer as recited in claim 47, wherein the request indicates that at least one subset of the objects comprise a similar attribute; and wherein the computer-executable instructions for accessing one or more objects further comprise instructions for:

retrieving the one or more objects in a manner that is independent of any hierarchical data relationship between the data objects in the at least one subset.

56. A computer as recited in claim 55, wherein the similar attribute comprises a logical domain selected from a distinguishing domain, a locating domain or a classifying domain.

57. A computer as recited in claim 47, wherein the request corresponds to at least a first and second subset of the objects, the request comprising a logical modifier that specifies an operation, and wherein the computer-executable instructions further comprise instructions for:

responsive to receiving the request, identifying at least a portion of the first and second subsets of directory objects in the polyarchical data set; and

wherein transforming the one or more objects further comprises performing the operation on the first and second subsets.

58. A computer as recited in claim 57, wherein the logical modifier is a Boolean modifier.

59. A computer as recited in claim 57, wherein the operation comprises any combination of filtering, union, intersection, join, and/or exclusion operations.

60. A computer as recited in claim 47, wherein accessing the one or more objects further comprises accessing the one or more objects in a manner that is independent of any inter-object relationship between the one or more objects and any other object of the objects in a manner that is independent of any definition of a hierarchy in the data polyarchy.

61. A computer as recited in claim 47, wherein accessing the one or more objects further comprises querying the data polyarchy for the one or more objects.

62. A computer as recited in claim 47, wherein accessing the one or more objects further comprises managing, manipulating, or modifying the one or more objects or a relationship between an object of the one or more objects and one of more different objects of the objects.

63. A computer comprising:

processing means for dynamically generating a schema to represent multiple hierarchies of inter-object relationships between a plurality of objects in a data polyarchy, the schema being generated based on values of attributes of the objects, the schema indicating each attribute of interest in the data polyarchy, the schema further indicating any of one or more dimensions of inter-object relationships within which objects that comprise at least a subset of the attributes of interest participate.

64. A computer as recited in claim 63, wherein the inter-object relationships comprise a flat relationship, a hierarchical relationship, and multiple intersecting hierarchies of relationships.

65. A computer as recited in claim 63, where the objects comprise enterprise resource planning (ERP) objects, directory based objects, or database objects.

66. A computer as recited in claim 63, further comprising processing means for communicating the schema to a client to indicate how the client is to interface with the objects in the data polyarchy.

67. A computer as recited in claim 63, further comprising processing means for:

receiving a request from a client based on the schema;

responsive to receiving the request:

accessing one or more objects in the data polyarchy based on the request;

transforming the one or more of the objects into transformed data that expresses any inter-object relationships based on the request; and

issuing the transformed data to the client.

68. A computer as recited in claim 67, wherein the transformed data expresses the inter-object relationships with respect to other objects in a same dimension or other

objects in a different dimension, the same and/or the different dimension being indicated by the request.

69. A computer as recited in claim 67, wherein the request comprises a limiting attribute to limit the transformed data by presenting the one or more objects only with respect to the limiting attribute.

70. A computer as recited in claim 67, wherein the request queries for information corresponding to an object in the data polyarchy with respect to one or more particular dimensions.

71. A computer as recited in claim 67, wherein the request comprises a dimension indicator to specify one or more hierarchies within which the data is to be presented in the transformed data.

72. A computer as recited in claim 67, wherein the request further comprises a distinguishing attribute, a classifying attribute, or a locating attribute.

73. A computer as recited in claim 67, wherein the request comprises a dimension information modifier to specify a particular direction and a particular depth to retrieve information from the data polyarchy.

74. A computer as recited in claim 73, wherein the dimension information modifier is a siblings indication to retrieve all objects with a same parent as a current object in the data polyarchy.

75. A computer as recited in claim 67, wherein the request indicates that at least one subset of the objects comprise a similar attribute; and wherein the means for accessing one or more objects further comprise means for:

retrieving the one or more objects in a manner that is independent of any hierarchical data relationship between the data objects in the at least one subset.

76. A computer as recited in claim 75, wherein the similar attribute comprises a logical domain selected from a distinguishing domain, a locating domain or a classifying domain.

77. A computer as recited in claim 67, wherein the request corresponds to at least a first and second subset of the objects, the request comprising a logical modifier that specifies an operation, and wherein the processing means further comprise means for:

responsive to receiving the request, identifying at least a portion of the first and second subsets of directory objects in the polyarchical data set; and

wherein transforming the one or more objects further comprises performing the operation on the first and second subsets.

78. A computer as recited in claim 77, wherein the logical modifier is a Boolean modifier.

79. A computer as recited in claim 77, wherein the operation comprises any combination of filtering, union, intersection, join, and/or exclusion operations.

80. A computer as recited in claim 67, wherein the means for accessing the one or more objects further comprises means for accessing the one or more objects in a manner that is independent of any inter-object relationship between the one or more objects and any other object of the objects in a manner that is independent of any definition of a hierarchy in the data polyarchy.

81. A computer as recited in claim 67, wherein the means for accessing the one or more objects further comprises querying the data polyarchy for the one or more objects.

82. A computer as recited in claim 67, wherein the means for accessing the one or more objects further comprises means for managing, manipulating, or modifying the one or more objects or a relationship between an object of the one or more objects and one of more different objects of the objects.

83. A polyarchical query language data structure comprising:

a first data field to specify a particular schema, the particular schema indicating how to meaningfully present or manage a plurality of objects in a data polyarchy based on values of attributes in the objects; and

a second data field to indicate an attribute of interest; and

a third data field to indicate how one or more objects comprising the attribute of interest are to be presented or managed with respect to one or more participating dimensions of inter-object relationships which are based on the schema.

84. A polyarchical query language data structure as recited in claim 83 further comprising a fourth data field to indicate a physical access strategy with respect to the data polyarchy, the physical access strategy being identified by indicating that the attribute of interest belongs to a distinguishing domain, a classifying domain, or a locating domain.

85. A polyarchical query language data structure as recited in claim 83, wherein the third data field further comprises a modifier to limit the one or more objects.

86. A polyarchical query language data structure as recited in claim 83, wherein the third data field further comprises a logical modifier to limit the one or more objects.

87. A polyarchical query language data structure as recited in claim 83, wherein the third data field further comprises a dimension information indicator for specifying a dimension within which to present the one or more objects.

88. A polyarchical query language data structure as recited in claim 83, wherein the third data field further comprises a dimension information indicator for specifying a particular direction and a particular depth within which to present a data relationship between a complex object of the one or more objects and one or more different objects of the one or more objects.

89. A polyarchical query language data structure as recited in claim 83, wherein each data field is expressed in an XML data format.

90. A polyarchical query language data structure as recited in claim 83, wherein the particular schema provides access to only a first subset of the objects to provide access control to the objects.

91. A computer-readable medium comprising a polyarchical query language data structure as recited in claim 83.

* * * * *

46/3,K/5 (Item 5 from file: 350) **Links**
Derwent WPIX

010635979 **Image available**
WPI Acc No: 1996-132932/**199614**
XRPX Acc No: N96-111846
Reference condition integrating device for database search operation - has database reference execution part which, on receiving database integrating condition specified by user, refers to database
Patent Assignee: NEC CORP (NIDE)
Inventor: NIHEI K
Number of Countries: 002 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 8016616	A	19960119	JP 94153198	A	19940705	199614	B
US 5819253	A	19981006	US 95498778	A	19950705	199847	

Priority Applications (No Type Date): JP 94153198 A 19940705
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 8016616	A		6	G06F-017/30	
US 5819253	A			G06F-017/30	

has database reference execution part which, on receiving database integrating condition specified by user, refers to database

Abstract (Basic): condition input parts (1). Using these input parts, different database reference condition types such as **SQL**, **menu**, natural language type etc are input to the device. If needed, the **user** is able to **select** and **combine** the different database reference conditions. The **combined** condition is input by the **user** to the device using an integrated reference condition receptionist part (2). The integrated reference condition...

...Any **change** in the integrated reference conditions stored in the memory part is performed by the **user** at any time. The integrated reference condition generation part (4) then performs the actual **combination** or integration of the database reference conditions as per the **user**'s specification stored in the memory part. This reference condition is then conveyed to an...

...of the execution part is displayed on the display part of the device for the **user**.

Title Terms: **USER**;
International Patent Class (Main): **G06F-017/30**

International Patent Class (Additional): **G06F-012/00**
Manual Codes (EPI/S-X): **T01-J05B3**

US005819253A

United States Patent [19]

Nihei

[11] Patent Number: 5,819,253

[45] Date of Patent: Oct. 6, 1998

[54] **DATABASE RETRIEVAL SYSTEM HAVING A PLURALITY OF INDEPENDENT INPUT MODES FOR DATABASE RETRIEVAL CONDITIONS**

[75] Inventor: **Katsumi Nihei**, Tokyo, Japan

[73] Assignee: **NEC Corporation**, Tokyo, Japan

[21] Appl. No.: **498,778**

[22] Filed: **Jul. 5, 1995**

[30] **Foreign Application Priority Data**

Jul. 5, 1994 [JP] Japan 6-153198

[51] **Int. Cl.**[6] **G06F 17/30**

[52] **U.S. Cl.** **707/2**; 707/1; 707/5; 707/7; 707/531; 364/962; 364/962.3; 364/963; 364/963.1; 364/963.3; 364/974; 364/974.4; 364/DIG. 2

[58] **Field of Search** ..... 395/600, 601; 707/5, 1, 7, 531; 364/962, 962.3, 963, 963.1, 963.3, 974, 974.4, DIG. 4, DIG. 2

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,332,014	5/1982	Nakazawa et al.	364/900
5,088,052	2/1992	Spielman et al.	395/158
5,140,692	8/1992	Morita	395/600
5,265,076	11/1993	Ohyama	369/28
5,278,980	1/1994	Pedersen et al.	707/2
5,297,042	3/1994	Morita	364/419.19
5,307,266	4/1994	Hayasaki et al.	364/419.07
5,347,623	9/1994	Takano et al.	395/157
5,398,338	3/1995	Yoshida	395/600
5,418,946	5/1995	Mori	392/600
5,497,489	3/1996	Menne	395/600
5,535,382	7/1996	Ogawa	395/600
5,592,663	1/1997	Nagamori	395/605

OTHER PUBLICATIONS

Oracle SQL *Plus, NEC Corporation, pp. 1–1–3–1.

Microsoft Access User's Guide, Microsoft, pp. 217–245.

Arita et al., Multi-modal Natural Language Interface MM–Simpla, NEC Corporation, 1993, pp. 585–589.

Uemura, Basics of Database System, Ohmsha, Ltd., pp. 92–167.

Primary Examiner—Thomas G. Black

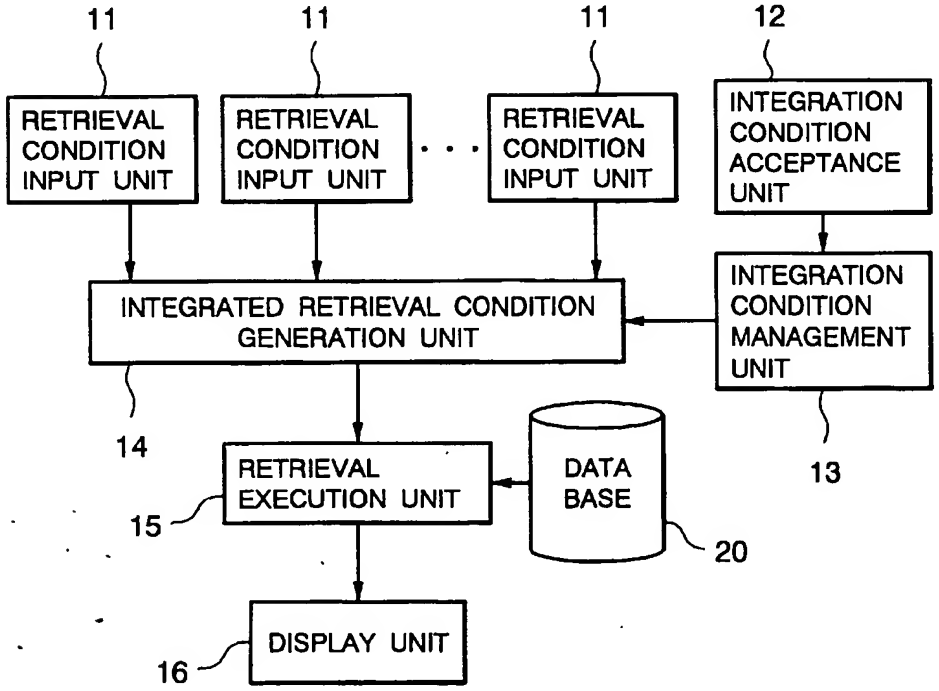
Assistant Examiner—Jean H. Corriélus

Attorney, Agent, or Firm—Foley & Lardner

[57] **ABSTRACT**

A database retrieval system includes a retrieval condition input unit for inputting database retrieval conditions in a plurality of retrieval modes different from each other, an integration condition acceptance unit for inputting integration conditions for combining a plurality of database retrieval conditions input through the retrieval condition input unit, an integrated retrieval condition generation unit for integrating retrieval conditions input by the retrieval condition input unit according to integration conditions input by the integration condition acceptance unit, and a retrieval execution unit for retrieving a database according to retrieval conditions integrated by the integrated retrieval condition generation unit.

16 Claims, 4 Drawing Sheets



expression nor an integration condition corresponding to the retrieval mode. The foregoing processing will be repeated until information corresponding to all the retrieval mode selection designation units **22** is read (Step **405**). Thereafter, the unit **14** combines the stored retrieval conditional expressions based on the stored integration conditions to generate and output an integrated retrieval conditional expression (Step **406**).

For example, when integration conditions are set as shown in FIG. **2**, an integrated retrieval conditional expression is generated that is a retrieval conditional expression of the retrieval mode a, a retrieval conditional expression of the retrieval mode b, and a retrieval conditional expression of the retrieval mode c.

The retrieval execution unit **15**, is implemented by program-controlled CPU and memory, or the like, of a personal computer or a work station. The execution unit **15** retrieves the database **20** according to an integrated retrieval conditional expression generated by the integrated retrieval condition generation unit **14** to output retrieval results. In the present invention conventional manners are applicable, but no specific manner of database retrieval is proposed here. For example, the method recited in the reference literature by Shunsuke Uemura, *Basics of Database System*, Ohmsha, Ltd., can be used.

The display unit **16** is implemented by a display device or the like. The display unit **16** displays retrieval results obtained by the retrieval execution unit **15**. The unit **16** is also capable of accommodating the interface panel **21** of the integration condition acceptance unit **12** as described above.

With reference to the flow chart shown in FIG. **5**, the flow of data retrieval processing according to the present embodiment will be described.

Referring to FIGS. **1**, **2**, and **5**, first, a user inputs a desired retrieval condition to the retrieval condition input unit **11** by using an input device such as a keyboard (Step **501**). Then, the user inputs a desired integration condition to the integration condition acceptance unit **12** by means of the interface panel **21** or the like (Step **502**). Upon input of these conditions, the integrated retrieval condition generation unit **14** receives input of a retrieval conditional expression from the retrieval condition input unit **11** and receives input of the integration condition from the integration condition management unit **13** to generate an integrated retrieval conditional expression (Step **503**). Then, based on the generated integrated conditional expression, the retrieval execution unit **15** executes retrieval of the database **20** (Step **504**) and the display unit **16** displays retrieval results obtained by the retrieval execution unit **15** (Step **505**).

Referring to the retrieval results displayed on the display unit **16**, the user is allowed to modify integration conditions when finding the retrieval results to be unsatisfactory (Step **506**). If the retrieval results are unsatisfactory then it goes back to Step **502**, and the user re-enters the integration conditions. When the integration conditions are re-entered, the integration condition management unit **13** detects modification of integration conditions so as to notify the integrated retrieval condition generation unit **14**. The integrated retrieval condition generation unit **14** generates an integrated retrieval conditional expression based on the new integration conditions in response to the notification from the integration condition management unit **13** (Step **503**). Then, the database **20** is retrieved (Step **504**) and retrieval results are displayed (Step **505**).

As described in the foregoing, since the database retrieval system of the present invention retrieves a database based on combined retrieval conditions in a plurality of different retrieval modes, entry of necessary retrieval conditions and their combinations enables the system to cope with the retrieval under multiple retrieval conditions, whereby time and labor necessary for the processing can be reduced.

In addition, referring to displayed retrieval results, a user is allowed to obtain retrieval results one after another based upon different integration conditions. It is therefore possible for the system of the present invention to execute flexible retrieval with ease according to a user's request.

Although the invention has been illustrated and described with respect to exemplary embodiment thereof, it is understood by those skilled in the art that the foregoing and various other changes, omissions and additions may be made therein and thereto, without departing from the spirit and scope of the present invention. Therefore, the present invention is not limited to the specific embodiment set out above, but is comprised of all possible embodiments which are within the scope, and equivalents thereof, with respect to the features set out in the appended claims.

What is claimed is:

1. A database retrieval system comprising:

a plurality of independent retrieval condition input units, each input unit receiving an input of database retrieval conditions and operating in a unique one of a first arbitrary number of different retrieval modes, each input unit generating a retrieval conditional expression based on the input, the number of input units corresponding to the first arbitrary number of retrieval modes;

integration condition input unit for inputting integration conditions for combining the retrieval conditional expressions generated by said independent retrieval condition input units;

retrieval condition integrating unit for integrating a second arbitrary number of the retrieval conditional expression s generated by said independent retrieval condition input units according to the integration conditions, said retrieval condition integrating unit generating an integrated retrieval conditional expression; and

retrieval execution unit for executing retrieval of a database according to the integrated retrieval conditional expression.

2. The database retrieval system as set forth in claim **1**, wherein the integration conditions include a condition for designating whether or not retrieval in a predetermined retrieval mode is to be executed and a logical operator indicative of a combination relationship among the retrieval conditional expressions associated with the retrieval modes designated to be executed.

3. The database retrieval system as set forth in claim **2**, wherein said retrieval condition integrating unit

refers to the designated condition of the retrieval modes, receives the retrieval conditional expressions associated with the retrieval modes designated to be executed from said retrieval condition input units, and stores the retrieval conditional expressions;

receives and stores said logical operator; and

after receiving all retrieval conditional expressions associated with the retrieval modes designated to be executed, combines the stored retrieval conditional expressions using the logical operator to generate the integrated retrieval conditional expression.

4. The database retrieval system as set forth in claim **1**, wherein the retrieval modes include at least one of SQL, a menu interface, and a natural language interface.

5. The database retrieval system as set forth in claim 1, further comprising:

integration condition management unit connected to receive the integration conditions from said integration condition input unit, and to notify said retrieval condition integrating unit when the integration conditions have been modified from previous integration conditions received from said integration condition input unit.

6. A retrieval condition integrating device in a database retrieval system comprising:

integration condition input unit for inputting integration conditions for combining a plurality of retrieval conditional expressions to generate an integrated retrieval conditional expression, each retrieval conditional expression being generated using a unique one of a first arbitrary number of different retrieval modes; and

retrieval condition integrating unit for integrating a second arbitrary number of the retrieval conditional expressions according to the integration conditions.

7. The retrieval condition integrating device in a database retrieval system as set forth in claim 6, wherein the integration conditions include a condition for designating whether or not retrieval in a predetermined retrieval mode is to be executed and a logical operator indicative of a combination relationship among the retrieval conditional expressions associated with the retrieval modes designated to be executed.

8. The retrieval condition integrating device in a database retrieval system as set forth in claim 6, wherein said retrieval condition integrating unit

refers to the designated condition of the retrieval modes, receives the retrieval conditional expressions associated with the retrieval modes designated to be executed, and stores the retrieval conditional expressions;

receives and stores said logical operator; and

after receiving all retrieval conditional expressions associated with the retrieval modes designated to be executed, combines the stored retrieval conditional expressions using the logical operator to generate the integrated retrieval conditional expression.

9. The retrieval condition integrating device in a database retrieval system as set forth in claim 6, wherein the retrieval modes include at least one of SQL, a menu interface, and a natural language interface.

10. The retrieval condition integrating device in a database retrieval system as set forth in claim 6, further comprising:

integration condition management unit connected to receive the integration conditions from said integration condition input unit, and to notify said retrieval condition integrating unit when the integration conditions have been modified from previous integration conditions received from said integration condition input unit.

11. A database retrieval method comprising the steps of:

(a) in putting database retrieval conditions and generating retrieval conditional expressions based on the database retrieval conditions, each retrieval conditional expression being generated using a unique one of a first arbitrary number of different retrieval modes;

(b) inputting integration conditions for combining a second arbitrary number of the retrieval conditional expressions generated in said step (a);

(c) integrating the retrieval conditional expressions generated in said step (a) according to the integration conditions to generate an integrated retrieval conditional expression; and

(d) retrieving a database according to the integrated retrieval conditional expression.

12. The database retrieval method as set forth in claim 11, wherein said step (b) of inputting integration conditions comprises the steps of:

designating whether or not retrieval in a predetermined retrieval mode is to be executed; and

setting a logical operator indicative of a combination relationship among the retrieval conditional expressions associated with the retrieval modes designated to be executed.

13. The database retrieval method as set forth in claim 12, wherein said step (c) of integrating retrieval conditions comprises the steps of:

with reference to the designated condition of a retrieval mode receiving and storing the retrieval conditional expressions associated with the retrieval modes designated to be executed;

receiving and storing said logical operator; and

after receiving all retrieval conditional expressions associated with the retrieval modes designated to be executed, combining the stored retrieval conditional expressions using the logical operator to generate the integrated retrieval conditional expression.

14. The database retrieval method as set forth in claim 11, wherein the retrieval modes include at least one of SQL, a menu interface, and a natural language interface.

15. A database retrieval system comprising:

at least four independent retrieval condition input units, each input unit receiving an input of database retrieval conditions and operating in a unique retrieval mode, each input unit generating a retrieval conditional expression based on the input;

integration condition input unit for inputting integration conditions for combining the retrieval conditional expressions generated by said independent retrieval condition input units;

retrieval condition integrating unit for integrating the retrieval conditional expressions generated by said independent retrieval condition input units according to the integration conditions, said retrieval condition integrating unit being capable of integrating more than three retrieval conditional expressions, and said retrieval condition integrating unit generating an integrated retrieval conditional expression; and

retrieval execution unit for executing retrieval of a database according to the integrated retrieval conditional expression.

16. A method of generating an integrated retrieval conditional expression comprising the steps of:

(a) providing an integration condition input unit having an arbitrary number of mode selection designation units corresponding to unique retrieval modes;

(b) inputting retrieval mode information into said mode selection designation units;

(c) inputting integration conditions corresponding to said retrieval mode information into said integration condition input unit;

(d) reading the retrieval mode information from one of said mode selection designation units into a retrieval condition integrating unit;

(e) determining whether the retrieval mode of said one of said mode selection designation units is to be used based on the retrieval mode information;

(f) if the retrieval mode is to be used, storing in said retrieval condition integrating unit a retrieval conditional expression corresponding to the retrieval mode, then reading the integration condition corresponding to said retrieval mode information from said integration condition input unit, and determining whether all integration conditions entered into said integration condition input unit have been read;

(g) if the retrieval mode is not to be used, determining whether all integration conditions entered into said integration condition input unit have been read;

(h) if all integration conditions entered into said integration condition input unit have been read, generating an integrated retrieval conditional expression based on the stored retrieval condition expressions and the stored integration conditions; and

(i) if all integration conditions entered into said integration condition input unit have not been read, repeating steps (d)–(g).

* * * * *

46/3,K/14 (Item 14 from file: 350) **Links**
Derwent WPIX

016947819 **Image available**
WPI Acc No: 2005-272127/200528
Related WPI Acc No: 2003-746922; 2005-272128
XRPX Acc No: N05-223543

Computer readable medium storing schema generation program, has specific data field to indicate presentation or management of objects comprising attribute of interest, with respect to participating dimensions of inter-object relationships

Patent Assignee: MICROSOFT CORP (MICT)
Inventor: BOOTH J H; CAMERON K; CLEMENT L; MACLEOD S P; ROBERTSON G G
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 20050071354	A1	20050331	US 2001995415	A	20011126	200528	B
			US 2004967435	A	20041018		

Priority Applications (No Type Date): US 2001995415 A 20011126; US 2004967435 A 20041018
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20050071354	A1		32	G06F-017/00	Cont of application US 2001995415

Computer readable medium storing schema generation program, has specific data field to indicate presentation or management of objects comprising attribute of interest, with respect to participating dimensions of inter-object relationships

Abstract (Basic):

... A polyarchical **query language** data structure has a data **field** to **specify** schema indicating meaningful presentation or management of **objects** in data polyarchy based on attribute values in **objects**. Other data **fields** indicate an attribute of interest, and indicate presentation or management of **objects** comprising the attribute of interest with respect to participating dimensions of inter-**object relationships** which are based on schema.

... For storing program of dynamically generated schema representing several hierarchies of inter-**object relationships** for computing device (claimed) e.g. personal computer (PC), server computer, thin **client**, thick **client**, hand-held or laptop device, multiprocessor system, microprocessor-based system, **set** top box, programmable consumer electronics, network PC, minicomputer and mainframe computer, in distributed computing environment...

...The inter-**object relationships** are managed and navigated, efficiently using data polyarchy schema that is generated to access elements...

...Title Terms: **FIELD**;

International Patent Class (Main): **G06F-017/00**

Manual Codes (EPI/S-X): **T01-F07**...

...**T01-J05B2B**...

...**T01-J05B3**...

...**T01-S03**

US 20050071354A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0071354 A1

Cameron et al. (43) Pub. Date: Mar. 31, 2005

(54) DYNAMICALLY GENERATED SCHEMA REPRESENTING MULTIPLE HIERARCHIES OF INTER-OBJECT RELATIONSHIPS

(75) Inventors: Kim Cameron, Bellevue, WA (US); Stewart P. MacLeod, Woodinville, WA (US); George G. Robertson, Seattle, WA (US); James H. Booth, Issaquah, WA (US); Luc Clement, Sammamish, WA (US)

Correspondence Address:
LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

(73) Assignee: Microsoft Corporation, Redmond, WA

(21) Appl. No.: 10/967,435

(22) Filed: Oct. 18, 2004

Related U.S. Application Data

(63) Continuation of application No. 09/995,415, filed on Nov. 26, 2001.

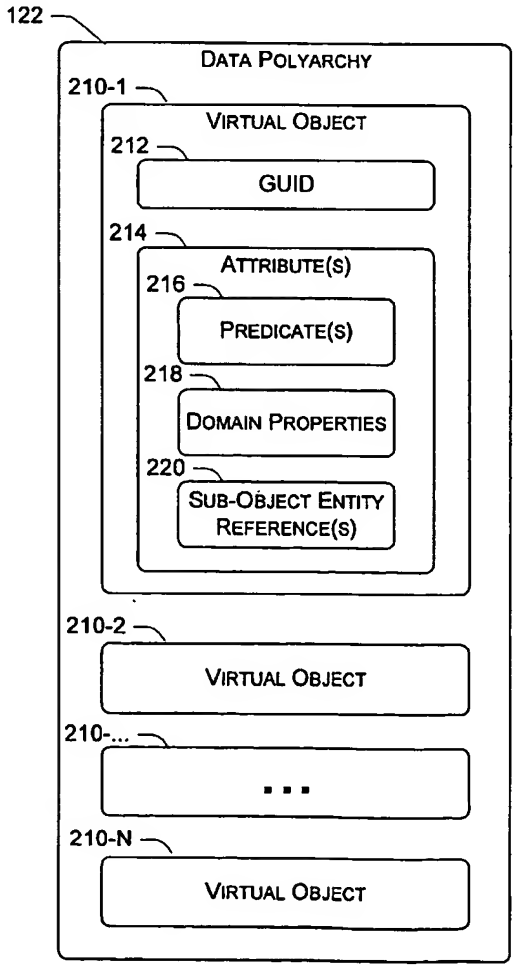
Publication Classification

(51) Int. Cl.[7] G06F 17/00

(52) U.S. Cl. 707/100

(57) ABSTRACT

Systems and methods for dynamically generating a schema representing multiple hierarchies of inter-object relationships are described. In one aspect, a polyarchical query language data structure includes first, second, and third data fields. The first data field is used to specify a particular schema for presenting or managing a plurality of objects in a data polyarchy based on values of attributes in the objects. The second data field is to indicate an attribute of interest. The third data field indicates how one or more objects that include the attribute of interest are to be presented or managed with respect to one or more participating dimensions of inter-object relationships based on the schema.



limited to the specific features or operations described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed present invention.

1. A computer-readable medium comprising:

a polyarchical query language data structure comprising:

a first data field to specify a particular schema, the particular schema indicating how to meaningfully present or manage a plurality of objects in a data polyarchy based on values of attributes in the objects; and

a second data field to indicate an attribute of interest; and

a third data field to indicate how one or more objects comprising the attribute of interest are to be presented or managed with respect to one or more participating dimensions of inter-object relationships which are based on the schema.

2. A computer-readable medium as recited in claim 1, wherein the third data field further comprises a modifier to limit the one or more objects.

3. A computer-readable medium as recited in claim 1, wherein the third data field further comprises a logical modifier to limit the one or more objects.

4. A computer-readable medium as recited in claim 1, wherein the third data field further comprises a dimension information indicator for specifying a dimension within which to present the one or more objects.

5. A computer-readable medium as recited in claim 1, wherein the third data field further comprises a dimension information indicator for specifying a particular direction and a particular depth within which to present a data relationship between a complex object of the one or more objects and one or more different objects of the one or more objects.

6. A computer-readable medium as recited in claim 1, wherein each data field is expressed in an XML data format.

7. A computer-readable medium as recited in claim 1, wherein the particular schema provides access to only a first subset of the objects to provide access control to the objects.

8. A computer-readable medium as recited in claim 1, wherein the polyarchical query language data structure further comprises a fourth data field to indicate a physical access strategy with respect to the data polyarchy, the physical access strategy being identified by indicating that the attribute of interest belongs to a distinguishing domain, a classifying domain, or a locating domain.

9. A method comprising:

specifying a particular schema via a polyarchical query language data structure, the polyarchical query language data structure comprising:

a first data field to specify the particular schema, the particular schema indicating how to meaningfully present or manage a plurality of objects in a data polyarchy based on values of attributes in the objects; and

a second data field to indicate an attribute of interest; and

a third data field to indicate how one or more objects comprising the attribute of interest are to be presented or managed with respect to one or more participating dimensions of inter-object relationships which are based on the schema.

10. A method as recited in claim 19, wherein the third data field further comprises a modifier to limit the one or more objects.

11. A method as recited in claim 19, wherein the third data field further comprises a logical modifier to limit the one or more objects.

12. A method as recited in claim 19, wherein the third data field further comprises a dimension information indicator for specifying a dimension within which to present the one or more objects.

13. A method as recited in claim 19, wherein the third data field further comprises a dimension information indicator for specifying a particular direction and a particular depth within which to present a data relationship between a complex object of the one or more objects and one or more different objects of the one or more objects.

14. A method as recited in claim 19, wherein each data field is expressed in an XML data format.

15. A method as recited in claim 19, wherein the particular schema provides access to only a first subset of the objects to provide access control to the objects.

16. A method as recited in claim 19, wherein the inter-object relationships comprise a flat relationship, a hierarchical relationship, and multiple intersecting hierarchies of relationships.

17. A method as recited in claim 19, where the objects comprise enterprise resource planning (ERP) objects, directory based objects, or database objects.

18. A method as recited in claim 19, wherein the polyarchical query language data structure further comprises a fourth data field to indicate a physical access strategy with respect to the data polyarchy, the physical access strategy being identified by indicating that the attribute of interest belongs to a distinguishing domain, a classifying domain, or a locating domain.

19. A computing device comprising:

a processor; and

a memory coupled to the processor, the memory comprising computer-program instructions executable by the processor for:

specifying a particular schema via a polyarchical query language data structure, the polyarchical query language data structure comprising:

a first data field to specify the particular schema, the particular schema indicating how to meaningfully present or manage a plurality of objects in a data polyarchy based on values of attributes in the objects; and

a second data field to indicate an attribute of interest; and

a third data field to indicate how one or more objects comprising the attribute of interest are to be presented or managed with respect to one or more participating dimensions of inter-object relationships which are based on the schema.

20. A computing device as recited in claim 19, wherein the third data field further comprises a dimension information indicator for specifying a dimension within which to present the one or more objects.

* * * * *

46/3,K/43 (Item 43 from file: 350) **Links**
Derwent WPIX

015594631 **Image available**
WPI Acc No: 2003-656786/200362
XRPX Acc No: N03-523210
Multimedia objects updating method, involves accepting agent program to invoke structured query language procedure to manipulate multimedia object in particular column and in particular table
Patent Assignee: ORACLE INT CORP (ORAC-N)
Inventor: DIAMOND D L; JANOSIK J L; OXBURY S J; RUBINO M
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6591295	B1	20030708	US 99434243	A	19991105	200362	B

Priority Applications (No Type Date): US 99434243 A 19991105
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6591295	B1		11	G06F-015/16	

Multimedia objects updating method, involves accepting agent program to invoke structured query language procedure to manipulate multimedia object in particular column and in particular table

Abstract (Basic):
... The method involves executing **graphical user interface** clipboard program (117) that accepts an agent program (111) **designation** to provide access to **relational** database (113). An alphanumeric uniform resource locator is transferred to **designate** a structured **query language** (**SQL**) procedure (112) to an application program. The **SQL** procedure is invoked using the locator to manipulate a multimedia **object**.
... An INDEPENDENT CLAIM is also included for an apparatus for storing, **updating** and retrieving data into a web server database
...

...Used for storing, retrieving and manipulating multimedia data **objects**.
...

...The clipboard program provides a **graphical user interface** that easily identifies and retrieves a multimedia **object** stored in the database. The clipboard is used to capture a multimedia **object** from a device through Internet to store the

new **object** in a **selected** row and column location in a **selected** database table, thereby facilitating the use of **objects** in web applications...

...The drawing shows a block diagram explaining the interactions between the principle components of multimedia **objects** storing and **updating** apparatus...

...**SQL** procedure (112...

...**Relational** database (113

Title Terms: **OBJECT;**
International Patent Class (Main): **G06F-015/16**
Manual Codes (EPI/S-X): **T01-J10**

US006591295B1

(12) **United States Patent**
Diamond et al.

(10) **Patent No.: US 6,591,295 B1**
(45) **Date of Patent: Jul. 8, 2003**

(54) **METHODS AND APPARATUS FOR USING MULTIMEDIA DATA STORED IN A RELATIONAL DATABASE IN WEB APPLICATIONS**

(75) Inventors: **David Lane Diamond**, Merrimack, NH (US); **John Louis Janosik, Jr.**, Nashua, NH (US); **Simon John Oxbury**, Nashua, NH (US); **Michael Rubino**, Nashua, NH (US)

(73) Assignee: **Oracle International Corp.**, Redwood Shores, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/434,243**

(22) Filed: **Nov. 5, 1999**

(51) **Int. Cl.**[7] **G06F 15/16**

(52) **U.S. Cl.** **709/217**; 709/219; 709/203; 709/226; 707/3; 707/10; 707/102

(58) **Field of Search** ..... 709/217, 218, 709/219, 226, 203; 707/102, 100, 3, 10, 10 G

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,924,099 A	*	7/1999	Guzak et al.	705/75
5,930,786 A	*	7/1999	Carino et al.	703/3
6,012,067 A	*	1/2000	Sarkar	358/1.13
6,260,044 B1	*	7/2001	Nagral et al.	707/10
6,397,219 B2	*	5/2002	Mills	707/10
6,400,378 B1	*	6/2002	Snook	345/716

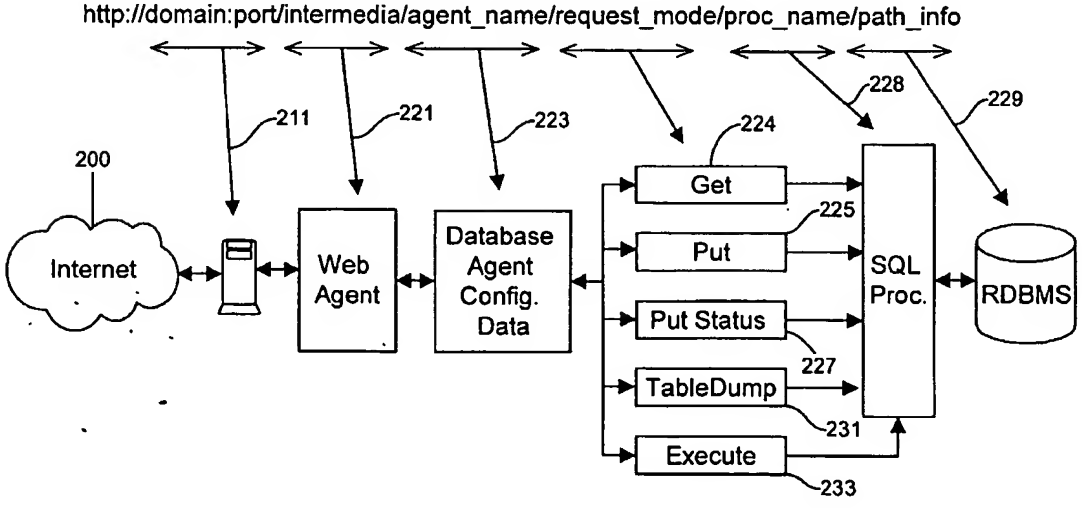
* cited by examiner

Primary Examiner—Ario Etienne
Assistant Examiner—Abdullahi E. Salod
(74) *Attorney, Agent, or Firm*—Sanjay Prasad

(57) **ABSTRACT**

A Web agent software interface between a relational database and Web-based application programs employs stored SQL procedures to store, update and retrieve multimedia objects. Web based applications manipulate the data in the relational database table by using URLs consisting of an alphanumeric designation of the host computer and port, the Web agent program associated with the database, the database agent configuration information associated with the database, specific stored SQL procedure for performing the desired data manipulation, and additional parameter data needed to specify and manipulate a particular multimedia object. A clipboard application program provides a graphical user interface which a user may employ to select and view selected multi-media objects stored in selected relational database tables, and to transfer the Web URLs designating those selected objects to other Web applications, such as HTML authoring tools, using convenient "drag-and-drop" and "cut, copy and paste" operations. The Web agent program retrieves data from the database which are designated by such URLs, and accepts POST requests from HTML forms which are activated by the user of a Web browser to load or update data in the relational database.

10 Claims, 4 Drawing Sheets



of the object is displayed and this value is passed via a parameter in the SQL procedure. The object is loaded into the database for persistent storage when user clicks on "Update" in the Database tab.

An existing multimedia object may updated in much the same way. The particular table containing the object is selected using the navigation tree **311**, and a row and column is selected in that table using the table view **315** so that the object to be updated is displayed in the object view. The table view may be used to display a subset of the records in any table sorted in a desired order using conventional database query and sort procedures selected via the menu at **310**. Once a particular object has been identified in a particular row and column location in the table view, that object may then be copied and pasted into an editor, saved as a file and opened in an editor, or replaced by a newly captured image or by a different image fetched via the Internet as described above. A list of one or more editors which may be invoked is presented at the Edit tab **338**. When the updated image is present in the object view, the Update button at the Database tab **337** is clicked to update the object.

To retrieve an object from the database and use that object in an application that supports "drag-and-drop" operations, the object is first displayed in the object view as described above. The user then specifies Table→Drag Mode→Embed from the menu at **310** to place a reference (URL) to the database object in the target application, such as an HTML editor for creating Web pages. The necessary <img src= "url"> tag including the database URL is then automatically inserted into the Web page HTML, and the image is seen in the Web page editor. Alternatively, by selecting "Edit→Drag Mode→Link" a link of the form "anchor text" containing the object's database URL is automatically placed in the Web page HTML. Having selected the drag mode, the icon which represents the object in the table view as seen at **341** may be dragged to and dropped into the target application, such as the Web page authoring tool which constructs the appropriate HTML tag.

Storing Data From Web Pages

The Web agent illustrated at **111** in FIG. **1** may also be used to store data supplied via a Post request method from an HTML form. When the HTML Web page containing the input form is written, the parameters needed are incorporated into the form markup as illustrated by the following example:

```
<form
action="http://nedc.us.ora.com:8007/intermedia/emp_write/mediaput/
PUT_EMP_PIC"
method="post" enctype="multipart/form-data">
<input type="hidden" name="ord_post_put_call" value=
"SET_EMP_PIC">
Employee ID: <input type="text" name="ord_procedure_path"
length=5>
Employee Picture File: <input type="file" name="ord_content">
<input type="Submit" value="Store Employee Picture Now">
</form>
```

FIG. **4** of the drawings illustrates an HTML web page that contains the form specified by the markup above. Note in the HTML above that the "action" specifies the full URL pathname for the SQL procedure PUT_EMP_PIC which has been previously stored for use by the Web agent in storing multimedia objects in a particular database table. In the URL, "emp_write" is the database agent name. The encoding type is specified as "multipart/form-data" and the multimedia object is identified with the variable name "ord_content" which specifies the value entered by the browser user for the particular image file that contains the multimedia object to be loaded. The key value of the table row which is to receive the object is specified by the "ord_procedure_path" variable name, and the "ord_post_put call" variable name is used to set the properties of the data after it is stored in the database. The variable specifies the name ("SET_EMP_PIC") of the SQL procedure that performs post-processing.

Using the method illustrated above, multimedia objects may be inserted or updated into a Web server database from any Web browser. Similarly, as noted earlier, multimedia objects may be imbedded in, or linked from, a Web page display using conventional Web page authoring tools in combination with the automatic URL generation and transfer mechanism provided by the clipboard and Web agent.

Conclusion

It is to be understood that the methods and apparatus which have been described are merely illustrative applications of the principles of the invention. Numerous modifications may be made by those skilled in the art without departing from the true spirit and scope of the invention.

What is claimed is:

1. A method of storing and updating multimedia objects in a relational database and retrieving said objects via the Internet comprising, in combination, the steps of:

connecting a host computer to the Internet,

programming said host computer to function as a relational database capable of storing, updating and retrieving data objects logically stored in one or more database tables,

executing an application program which utilizes multimedia objects,

executing a graphical user interface clipboard program for performing the steps of:

accepting from a user the designation of an agent program which provides access to said relational database, the designation of a particular table in said database, and the designation of a particular column in said particular table,

accepting from said user and storing a SQL procedure at said host computer, said SQL procedure being expressed at least in part in a structured query language for manipulating a multimedia object in said particular column in said particular table, and transferring an alphanumeric Uniform Resource Locator which designates said SQL procedure from said clipboard program to said further application program, and

executing said agent program for performing the steps of:

accepting a data manipulation request from said application program which includes said alphanumeric Internet Uniform Resource Locator, and

invoking said SQL procedure designated by said alphanumeric Uniform Resource Locator to manipulate said multimedia object.

2. The method set forth in claim **1** further comprising the steps of:

placing a hypertext link in a Web page, said link including said Uniform Resource Locator, and

processing said Web page in a browser application program to activate said hypertext link by transmitting said data manipulation request for processing by said agent program.

3. The method of claim 1 wherein said step of transferring is performed using an operating system interprocess data transfer procedure.

4. The method of claim 1 wherein said step of transferring is performed by executing a drag-and-drop procedure in which an iconic representation of said multimedia object is visually moved between a screen display area produced by said clipboard and a screen display area produced by said application program.

5. The method of claim 3 wherein transfer procedure is an operating system copy and paste procedure.

6. The method of claim 3 wherein said transfer procedure is an operating system cut and paste procedure.

7. Apparatus for storing, updating and retrieving data comprising, in combination,

a host computer including a memory for storing programs and data,

a relational database program stored in and executable by said host computer for storing, updating and retrieving data in one or more database tables, said relational database program including means for interpreting and executing stored data manipulation procedures expressed at least in part in a structured query language,

means for storing at least one of said procedures for manipulating specific data stored in one or more of said database tables, said one of said procedures being designated by a procedure name and including means for processing externally supplied parameter information to locate said specific data in said database tables,

a Web agent program stored in and executable by said host computer,

a HTTP interface program for receiving data requests via the Internet from a remotely located computer directed to a resource specified by a multi-part alphanumeric Universal Resource Locator (URL) which consists of:

(a) an identification of the domain name and port of said host computer,

(b) the identification of said Web agent program which operates as an interface to a relational database,

(c) the specification of said one of said procedures, and

(d) said parameter information,

said Web agent program including means for invoking the operation of said database program to perform said one of said procedures to perform a predetermined data manipulation operation with respect to said specific data, and

a graphical interface application program stored in and executable by said host computer for transferring data from said relational database to a second application program executing on said host computer, said clipboard application program comprising:

means for displaying representations of the content of said database tables,

means for accepting from a user an identification of at least a selected one of said displayed representations to identify said specific data stored in one or more of said database tables, and

means for transferring to said second application program a Uniform Resource Locator which specifies one of said procedures and parameter information and may be transmitted to said interface program to invoke said Web agent program for retrieving said specific data from said database tables using said one of said procedures.

8. The apparatus of claim 7 wherein said means for transferring includes means for performing a drag-and-drop procedure in which an iconic representation of said specific data is visually moved between a screen display area produced by said clipboard application program and a screen display area produced by said second application program.

9. The apparatus of claim 8 wherein said means for transferring further includes means for performing a copy and paste procedure.

10. The apparatus of claim 9 wherein said means for transferring further includes means for performing a cut and paste procedure.

* * * * *

46/3,K/49 (Item 49 from file: 350) **Links**
Derwent WPIX

015441274 **Image available**
WPI Acc No: 2003-503416/200347
XRAM Acc No: C03-134486
XRPX Acc No: N03-399615

Biological data accessing system used for searching database of biological information comprises database graph generation module associated with search engine, and joins module to create joins between database tables based on graph

Patent Assignee: ROTH C (ROTH-I)
Inventor: ROTH C
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 20030041053	A1	20030227	US 2001938714	A	20010823	200347	B

Priority Applications (No Type Date): US 2001938714 A 20010823
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20030041053	A1		17	G06F-007/00	

... **searching database of biological information comprises database graph generation module associated with search engine, and joins module to create joins between database tables based on graph**

Abstract (Basic):

... database graph generation module associated with a search engine to generate a database graph and **joins** module to create **joins** between database tables based on database graph. The server computer runs a structured **query language** search on the database based upon the **joins**.

... graph generation module associated with a search engine to generate a database graph and a **joins** module to create **joins** between database tables based on the database graph. The server computer runs a structured **query language** (**SQL**) search on the database based upon the **joins**.

...

...An INDEPENDENT CLAIM is also included for querying a **relational** database which comprises sending a structured language database query to a search engine, parsing the database and creating a database graph, creating the correct **joins corresponding** to the query, translating the structured query into an **SQL statement** incorporating the **joins** and sending the **SQL statement** to a **relational** database...

...of time to rewriting the code for existing databases or components and to minimize the **changes** to existing databases needed to **update** the system with new functionalities. The system preserves the property of component reuse, and is...

...a new utility or application is added to the system, or if a component is **changed**, little or no **lines** of code have to be **changed** in other applications or services improving upgrade performance and reducing maintenance time

Technology Focus:

... **Preferred** Components: The system comprises a second module that receives the results of the **SQL** search and translates the search results into a structured language. The **SQL** is sent to a **client** computer. The first module comprises a **user interface** that provides a list of searchable **fields** within the database and comprises a viewer module to present search results in a graphical

...Title Terms: **JOIN;**

International Patent Class (Main): **G06F-007/00**

Manual Codes (EPI/S-X): **T01-E**

US 20030041053A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0041053 A1

Roth (43) Pub. Date: Feb. 27, 2003

(54) SYSTEM AND METHOD FOR ACCESSING BIOLOGICAL DATA

(76) Inventor: Chantal Roth, San Diego, CA (US)

Correspondence Address:
KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614 (US)

(21) Appl. No.: 09/938,714

(22) Filed: Aug. 23, 2001

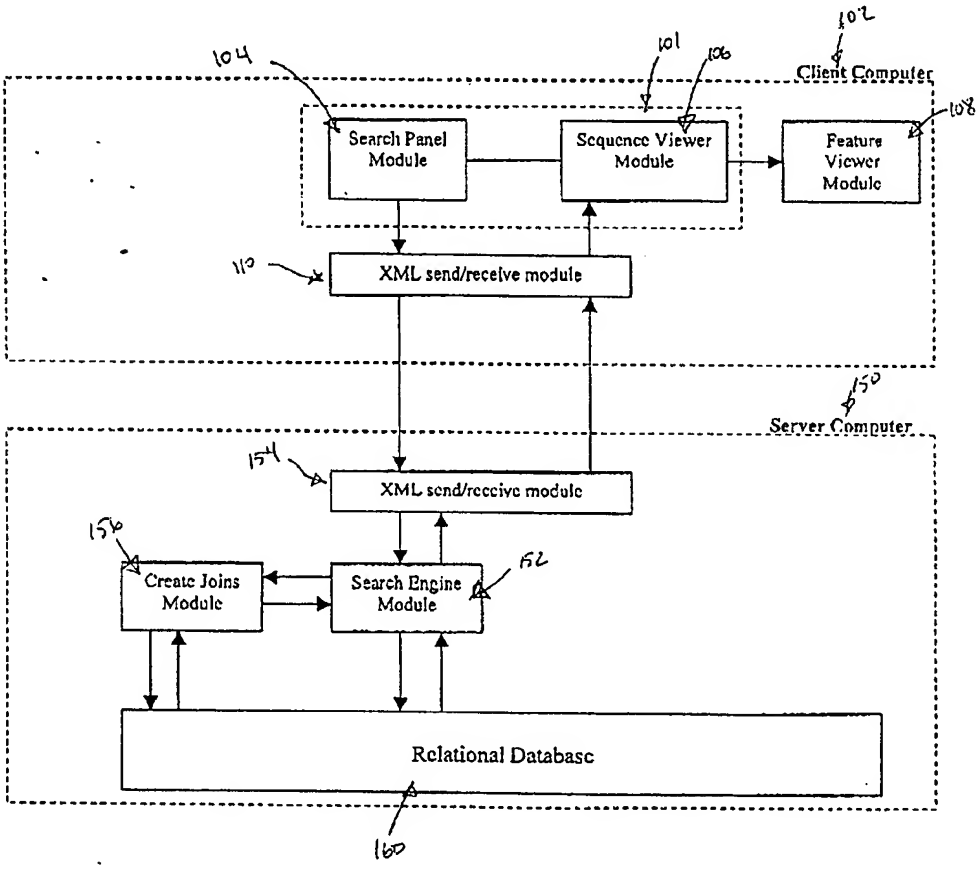
Publication Classification

(51) Int. Cl.[7] G06F 7/00

(52) U.S. Cl. 707/3

(57) ABSTRACT

A system is presented with a search panel for specifying search criteria for searching a database of biological information. The search panel uses the extensible markup language (XML) to send search requests to the database. A database graph generation module linked to the biological database generates a database graph representing the database schema. Once the database schema is known, another module creates joins between the database tables in order to most effectively join data from one table to another. An SQL statement incorporating the optimized joins is then used to search the biological database.



the Sequence Viewer Module **106** for representation to the user. The user can then view and manipulate the results, including invoking a Feature Viewer Module **108** to graphically represent the returned results.

[0090] This specific embodiment describes a method for searching a remotely located database. Searching a remote database in this manner is much slower than locating the Search Engine **520** on the Server Computer **150** along with the Relational Database **160**. In this embodiment, the Database Graph Generation Module **502** must communicate and receive a substantial amount of information with the Relational Database **160** through a network system. The time required for this exchange is controlled by the available bandwidth, which will be much slower than if the Search Engine **520** were able to communicate with a Relational Database **160** stored on the same computer. Hence, locating the Search Engine **520** on the Server Computer **150** allows the Database Graph Generation Module **502** to perform its instructions much faster than if it had to communicate across a network protocol such as a TCP protocol.

[0091] In reference to **FIG. 7**, a process flow diagram is depicted. A User **702** accessing a Client Computer utilizes the Search Panel Database Selector **202** field presented within the Search Panel User Interface **704** to choose a database stored on a Server Computer in which to search. Alternatively, a User **702** may specify a Server Computer or a folder stored on a Server Computer in which to conduct the search without specifying a specific database. This way, all databases contained within the specified folder or stored on the specified computer can be searched according to the search criteria, thus allowing a user to retrieve specific results without designating a particular database.

[0092] Once a search location is selected, the user specifies filtering criteria in the Search Panel User Interface **704** presented by the Search Panel Module. An XML File **706** containing the search criteria is created and sent to the Search Tool Module **506** of the Search Engine, which resides on a server computer. The Search Tool Module **506** instructs the Database Graph Generation Module **502** to create a database graph. This is done by parsing the database and accessing the Entry Definitions **710**. The Entry Definitions **710** have associated Data Entries **712**, which include at least one Primary Key **714** and may contain one or more Secondary Keys **716**. The database graph is generated, representing the database schema, from which the Join Path Generation Module **504** is able to construct the correct joins between the database tables and then subsequently calculate the most efficient path between the requested nodes. The efficient path is communicated to the Search Tool Module **506**, which then sends the appropriate query, including the requested nodes and the calculated joins between the nodes, to the SQL Statement Generation Module **508**. The SQL Statement Generation Module **508** translates the query into an appropriate SQL Statement **720** and sends the query to the Relational Database **160**, which is governed by database management software, and returns the requested information in a corresponding SQL Statement **722** to the Search Tool Module **506**.

[0093] The Search Tool Module **506** sends the SQL results to an XML send/receive module **154** which translates the SQL results into an appropriate XML file **726**. The XML send/receive module **154** then sends the XML Results File **726** to the Search Panel User Interface **704**. The graphical results are displayed to the User **702** in the Sequence Viewer Module **106**. The User **702** is then able to invoke an external Viewing Application **728** to further view and modify the returned results.

[0094] The foregoing description specifically describes one embodiment and method of practicing the current invention. The invention, however, is not limited to the embodiment described herein. It should be understood that changes may be made to the specific modules or information flow without departing from the scope of the invention, thus, the scope of the present invention is defined solely by the following claims.

What is claimed is:

1. A system for searching a database of biological information, said system comprising:

a server computer comprising a database of biological information and a first module for receiving a structured language query and transferring said query to a search engine;

a database graph generation module associated with said search engine configured to generating a database graph; and

a joins module configured to create joins between database tables based on said database graph, wherein said server computer runs a structured query language (SQL) search on said database based upon said joins.

2. The system of claim 1, comprising a second module that receives the results of said SQL search and translates said search results into a structured language.

3. The system of claim 2, wherein said structured query language is sent to a client computer.

4. The system of claim 1, wherein said first module comprises a user interface that provides a list of searchable fields within said database.

5. The system of claim 1, wherein said first module comprises a viewer module configured to present search results in a graphical format.

6. The system of claim 1, wherein said structured language comprises the extensible markup language (XML), JavaScript, or the hypertext markup language (HTML).

7. A computer system for searching a database of biological information, comprising:

a database of biological information comprising tables of biological data;

a search module configured to receive a structured language query and convert said structured language query into a search statement for querying said database of biological information; and

a joins module configured to determine how to join said tables of biological data in order to provide the results of said query.

8. The search engine of claim 7, further comprising in XML send/receive Module for sending and receiving information to and from a Search Panel Module stored on a Client Computer.

9. The search engine of claim 8, wherein said XML send/receive Module receives an XML structured query from a Client Computer, and delivers said XML structured query to a search tool module.

10. The search engine of claim 7, wherein said Database Graph Generation Module creates a graph of a user-selected database.

11. The search engine of claim 10, wherein said Create Joins Module utilizes said database graph to create joins between database tables.

12. The search engine of claim 11, wherein said Create Joins Module calculates the shortest path between two database nodes thereby optimizing the retrieval of requested database data.

13. The search engine of claim 7, further comprising a SQL statement generation module for translating said XML structured query into an SQL statement and sending said SQL statement to said Relational Database.

14. A method for querying a relational database, comprising the steps of:

sending a structured language database query to a search engine;

parsing the database and creating a database graph;

creating the correct joins corresponding to said query;

translating said structured database query into an SQL statement incorporating said joins; and

sending said SQL statement to a Relational Database.

15. The method of claim 14, including the further step of optimizing said joins by calculating the shortest path between the nodes specified in said query.

16. The method of claim 15, including the further step of receiving requested results from said database search, translating said results into said structured data language and returning said results.

17. The method of claim 16, including the further step of displaying said search results.

18. The method of claim 18, wherein said structured language is the Extensible Markup Language.

* * * * *

46/3,K/16 (Item 16 from file: 350) **Links**
Derwent WPIX

016798651 **Image available**
WPI Acc No: 2005-122930/200513
XRPX Acc No: N05-106056

Structured query language script generation method for database management system, involves selecting associated template strings based on each instruction selected corresponding to type of user defined objects in data model

Patent Assignee: DEFFLER T A (DEFF-I); COMPUTER ASSOC THINK INC (COMP-N)
Inventor: DEFFLER T A
Number of Countries: 108 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 200508534	A2	20050127	WO 2004US21990	A	20040709	200513 B
US 20050120014	A1	20050602	US 2003486773	P	20030711	200537
			US 2004888146	A	20040709	

EXACT SAME FILING DATE, DIFFERENT APPLICANT

Priority Applications (No Type Date): US 2003486773 P 20030711; US 2004888146 A 20040709
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 200508534	A2	E	24	G06F-017/30	

Designated States (National): AE AG AL AM AT AU AZ BA BB BG BR BW BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EC EE EG ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NA NI NO NZ OM PG PH PL PT RO RU SC SD SE SG SK SL SY TJ TM TN TR TT TZ UA UG US UZ VC VN YU ZA ZM ZW
Designated States (Regional): AT BE BG BW CH CY CZ DE DK EA EE ES FI FR GB GH GM GR HU IE IT KE LS LU MC MW MZ NA NL OA PL PT RO SD SE SI SK SL SZ TR TZ UG ZM ZW

US 20050120014 A1 G06F-017/30 Provisional application US 2003486773

Structured query language script generation method for database management system, involves selecting associated template strings based on each instruction selected corresponding to type of user defined objects in data model

Abstract (Basic):

... The associated template **strings** (150) are **selected** based on each instruction **selected corresponding** to type of **user** defined **objects** (142) **selected** in data **model** (140): A portion of structured **query language** (**SQL**) **script** (160) is generated automatically based on **selected** template **strings**.

... 1) structured **query language script**

generation program; and...

...2) structured **query language script** generation system...

...For generating structured **query language** (**SQL**) **script** in database management system (DBMS) in computer...

...Enables **customizing** the **text** generated in any suitable area of **SQL script**, and reduces debugging time and administration time of data...

...The figure shows a schematic diagram of the **SQL script** generation system...

...data **model** (140...

...**user** defined **objects** (142...

...template **strings** (150...

...**SQL script** (160
...Title Terms: **SCRIPT**;
International Patent Class (Main): **G06F-017/30**
Manual Codes (EPI/S-X): **T01-J05B3**...

...**T01-S03**

US 20050120014A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0120014 A1

Deffler (43) Pub. Date: Jun. 2, 2005

(54) SYSTEM AND METHOD FOR GENERATING SQL USING TEMPLATES

(76) Inventor: Tad A. Deffler, Boonton, NJ (US)

Correspondence Address:
FISH & RICHARDSON P.C.
5000 BANK ONE CENTER
1717 MAIN STREET
DALLAS, TX 75201 (US)

(21) Appl. No.: 10/888,146

(22) Filed: Jul. 9, 2004

Related U.S. Application Data

(60) Provisional application No. 60/486,773, filed on Jul. 11, 2003.

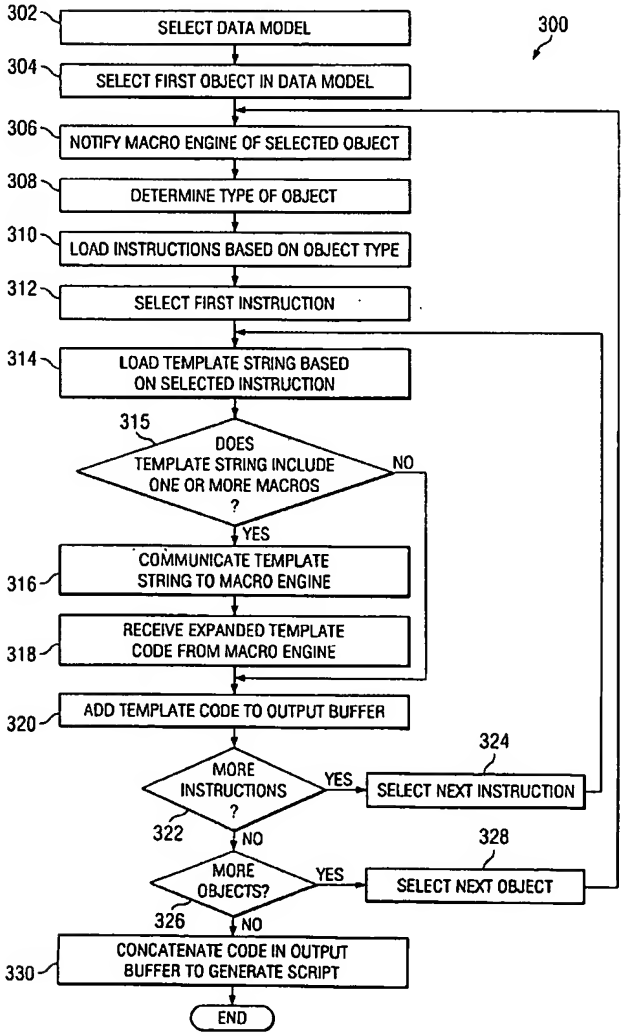
Publication Classification

(51) Int. Cl.[7] G06F 17/30

(52) U.S. Cl. 707/4

(57) ABSTRACT

A method for generating a structured query language (SQL) script based on a template includes selecting one object from a plurality of objects in a data model. At least one instruction is selected based, at least in part, on a type of the selected object and, then, an associated template string is selected based on each selected instruction. At least a portion of a SQL script is automatically generated based on the one or more selected template strings.



concatenate the portions of template code based, at least in part, on upon an order type (such as bucket identifier) stored in the template string, instruction, or object **142** as appropriate.

[0028] The preceding flowchart and accompanying description illustrate only an exemplary method **300** used by script generator **130** to generate SQL scripts **160** based on templates **150**. However, system **100** contemplates script generator **130** using any suitable technique for performing these tasks. Thus, many of the steps in this flowchart may take place simultaneously and/or in different orders than as shown. Moreover, script generator **130** may implement methods with additional steps, fewer steps, and/or different steps, so long as the methods remain appropriate.

[0029] Although this disclosure has been described in terms of certain embodiments and generally associated methods, alterations and permutations of these embodiments and methods will be apparent to those skilled in the art. Accordingly, the above description of example embodiments does not define or constrain this disclosure. Other changes, substitutions, and alterations are also possible without departing from the spirit and scope of this disclosure.

What is claimed is:

1. A method for generating a structured query language (SQL) script based on a template comprises:

selecting one object from a plurality of objects in a data model;

selecting at least one instruction based, at least in part, on a type of the selected object;

selecting an associated template string based on each selected instruction; and

automatically generating at least a portion of a SQL script based on the one or more selected template strings.

2. The method of claim 1, one or more of the plurality of objects comprising a user-defined object.

3. The method of claim 1 further comprising selecting the data model from a plurality of data models.

4. The method of claim 1 further comprising:

determining the existence of one or more macros in one of the selected template strings; and

in response, at least in part, to locating one or more macros in the template string, processing the located one or more macros.

5. The method of claim 4 further comprising expanding the template string associated with the one or more macros based on a result from each macro processing and wherein generating the SQL script is further based on the expanded template string.

6. The method of claim 4, each macro selected from the group comprising:

determining a value of an environment variable;

determining a value of a property associated with the data model;

determining a value of a property associated with the selected object;

evaluating a conditional expression; and

iterating over a subset of the plurality of objects, the subset associated with the selected object.

7. The method of claim 1 further comprising:

selecting a second object from the plurality of objects in the data model;

selecting at least one instruction based, at least in part, on a type of the second object;

selecting an associated template string based on each selected instruction;

concatenating the template strings from the selected objects; and

wherein generating the SQL script based on the one or more selected template strings comprises generating the SQL script based on the concatenated template strings.

8. The method of claim 7, each instruction comprising a template string identifier and a bucket identifier and the method further comprises sorting the concatenated template strings based on the bucket identifier of each associated instruction.

9. The method of claim 1 further comprising dynamically adding user comments to the SQL script based on at least one of the selected template strings.

10. Software for generating a structured query language (SQL) script based on a template operable to:

select one object from a plurality of objects in a data model;

select at least one instruction based, at least in part, on a type of the selected object;

select an associated template string based on each selected instruction; and

automatically generate at least a portion of a SQL script based on the one or more selected template strings.

11. The software of claim 10, one or more of the plurality of objects comprising a user-defined object.

12. The software of claim 10 further operable to select the data model from a plurality of data models.

13. The software of claim 10 further operable to:

determine the existence of one or more macros in one of the selected template strings; and

in response, at least in part, to locating one or more macros in the template string, process the located one or more macros.

14. The software of claim 13 further operable to expand the template string associated with the one or more macros based on a result from each macro processing and wherein generating the SQL script is further based on the expanded template string.

15. The software of claim 13, each macro selected from the group comprising:

determining a value of an environment variable;

determining a value of a property associated with the data model;

determining a value of a property associated with the selected object;

evaluating a conditional expression; and

iterating over a subset of the plurality of objects, the subset associated with the selected object.

16. The software of claim 10 further operable to:

select a second object from the plurality of objects in the data model;

select at least one instruction based, at least in part, on a type of the second object;

select an associated template string based on each selected instruction;

concatenate the template strings from the selected objects; and

wherein the software operable to generate the SQL script based on the one or more selected template strings comprises software operable to generate the SQL script based on the concatenated template strings.

17. The software of claim 16, each instruction comprising a template string identifier and a bucket identifier and the software further operable to sort the concatenated template strings based on the bucket identifier of each associated instruction.

18. The software of claim 10 further operable to dynamically add user comments to the SQL script based on at least one of the selected template strings.

19. A system for generating a structured query language (SQL) script based on a template comprises:

memory operable to store a data model, a plurality of instructions, and a plurality of template strings and the data model comprising a plurality of objects; and

one or more processors operable to:

select one object from the plurality of objects in the data model;

select at least one instruction based, at least in part, on a type of the selected object;

select an associated template string based on each selected instruction; and

automatically generate at least a portion of a SQL script based on the one or more selected template strings.

20. The system of claim 19, one or more of the plurality of objects comprising a user-defined object.

21. The system of claim 19, the one or more processors further operable to select the data model from a plurality of data models.

22. The system of claim 19, the one or more processors further operable to:

determine the existence of one or more macros in one of the selected template strings; and

in response, at least in part, to locating one or more macros in the template string, process the located one or more macros.

23. The system of claim 22, the one or more processors further operable to expand the template string associated with the one or more macros based on a result from each macro processing and wherein generating the SQL script is further based on the expanded template strings.

24. The system of claim 22, each macro selected from the group comprising:

determining a value of an environment variable;

determining a value of a property associated with the data model;

determining a value of a property associated with the selected object;

evaluating a conditional expression; and

iterating over a subset of the plurality of objects, the subset associated with the selected object.

25. The system of claim 19, the one or more processors further operable to:

select a second object from the plurality of objects in the data model;

select at least one instruction based, at least in part, on a type of the second object;

select an associated template string based on each selected instruction;

concatenate the template strings from the selected objects; and

wherein the processors further operable to generate the SQL script based on the one or more selected template strings comprises the processors further operable to generate the SQL script based on the concatenated template strings.

26. The system of claim 25, each instruction comprising a template string identifier and a bucket identifier and the one or more processors further operable to sort the concatenated template strings based on the bucket identifier of each associated instruction.

27. The system of claim 19, the one or more processors further operable to dynamically add user comments to the SQL script based on at least one of the selected template strings.

28. A system for generating a structured query language (SQL) script based on a template comprises:

means for selecting one object from a plurality of objects in a data model;

means for selecting at least one instruction based, at least in part, on a type of the selected object;

means for selecting an associated template string based on each selected instruction; and

means for automatically generating at least a portion of a SQL script based on the one or more selected template strings.

* * * * *

46/3,K/3 (Item 3 from file: 350) **Links**
Derwent WPIX

011030956 **Image available**
WPI Acc No: 1997-008880/**199701**
XRPX Acc No: N97-008090
Database structure conversion - involves structured-query-language sentence conversion from physical structure into logic structure figure on table definition information database
Patent Assignee: NEC CORP (NIDE)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 8278979	A	19961022	JP 9581518	A	19950406	199701	B

Priority Applications (No Type Date): JP 9581518 A 19950406
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 8278979	A		4	G06F-017/30	

Database structure conversion - ...
...**involves structured-query-language sentence conversion from physical structure into logic structure figure on table definition information database**

...Abstract (Basic): The method involves automatic bi-directional **updating** of computer system logic structure in database table definition information. The computer system logic structure is derived from a **physical** structure...

...The **physical** structure is retrieved from an external memory (23) by a structured-**query-language** decipher unit (11) constituted by an **SQL sentence** which is decomposed into several **physical sentence** components. The **sentence** components are **converted** into **logical** structures by a structure figure generator (12...
...Title Terms: **CONVERT**;

46/3,K/4 (Item 4 from file: 347) **Links**
JAPIO

05323479 **Image available**
METHOD AND DEVICE FOR DATA BASE STRUCTURE CONVERSION

Pub. No.: 08-278979 [JP 8278979 A]
Published: October 22, 1996 (**19961022**)
Inventor: HONDA MASAO
Applicant: NEC CORP [000423] (A Japanese Company or Corporation), JP (Japan)
Application No.: 07-081518 [JP 9581518]
Filed: April 06, 1995 (19950406)
METHOD AND DEVICE FOR DATA BASE STRUCTURE CONVERSION ...
Published: 19961022)

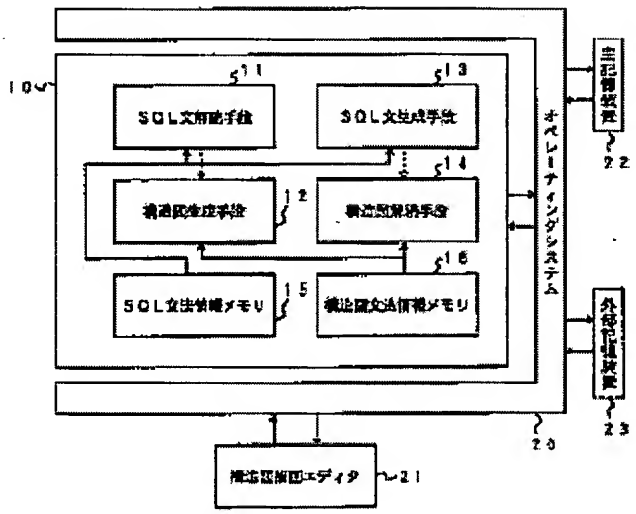
ABSTRACT
PURPOSE: To eliminate contradiction between the logical structure and physical structure of a data base by synchronizing both with each other... ...CONSTITUTION: The data base of a computer system is converted between the logical structure based upon grammar for the logical structure and the physical structure based upon grammar for the physical structure. Physical structure statements corresponding to the data base in the physical structure are stored in an external storage device 23. An SQL statement decoding means 11 reads a physical structure statement out of the external storage device 23 and decomposes the physical structure statement into plural physical structure statement elements by using the grammar for the physical structure. A structure diagram generating mean 12 converts the physical structure statement elements into logical structure statement elements. The logical structure statement elements are processed according to the grammar for the logical structure.

METHOD AND DEVICE FOR DATA BASE STRUCTURE CONVERSION

Patent number: JP8278979
Publication date: 1996-10-22
Inventor: HONDA MASAO
Applicant: NIPPON ELECTRIC CO
Classification:
- international: *G06F12/00; G06F17/30;* G06F12/00; G06F17/30; (IPC1-7): G06F17/30; G06F12/00
- european:
Application number: JP19950081518 19950406
Priority number(s): JP19950081518 19950406

Abstract of **JP8278979**

PURPOSE: To eliminate contradiction between the logical structure and physical structure of a data base by synchronizing both with each other. CONSTITUTION: The data base of a computer system is converted between the logical structure based upon grammar for the logical structure and the physical structure based upon grammar for the physical structure. Physical structure statements corresponding to the data base in the physical structure are stored in an external storage device 23. An SQL statement decoding means 11 reads a physical structure statement out of the external storage device 23 and decomposes the physical structure statement into plural physical structure statement elements by using the grammar for the physical structure. A structure diagram generating mean 12 converts the physical structure statement elements into logical structure statement elements. The logical structure statement elements are processed according to the grammar for the logical structure.



46/3,K/125 (Item 125 from file: 350) **Links**
Derwent WPIX

THE APPLICANT

009460710 **Image available**
WPI Acc No: 1993-154237/199319
XRPX Acc No: N93-117982

Graphical data base query system - includes selector for set of displayed parameters in order to define required query

Patent Assignee: INT BUSINESS MACHINES CORP (IBMC)
Inventor: BANNING K R; JAMES W S; LI S
Number of Countries: 004 Number of Patents: 003
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 541298	A2	19930512	EP 92309943	A	19921029	199319	B
EP 541298	A3	19931215	EP 92309943	A	19921029	199514	
US 5421008	A	19950530	US 91789507	A	19911108	199527	

Priority Applications (No Type Date): US 91789507 A 19911108
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 541298	A2	E	42	G06F-015/403	
Designated States (Regional): DE FR GB					
US 5421008	A		40	G06F-015/40	
EP 541298	A3			G06F-015/403	

... **includes selector for set of displayed parameters in order to define required query**

...Abstract (Basic): The apparatus for directing queries to a database containing information in order to access **selected** information comprises a display (38) for displaying in graphical form a number of parameters relating to queries (51 to 57). A **selection** unit (26) **selects** a **set** of the displayed parameters in order to define a required query and an operating unit is responsive to the **selection** of the parameters for performing the required query...

...A number of parameters relating to the query are displayed graphically. A **set** of parameters are **selected** in order to define the query. The method includes defining **relationships** between the **selected** parameters and defining **logical relationships** between them also...

...USE - Graphical database for structured **query language** (**SQL**) database systems...

...Abstract (Equivalent): extracting data from a first database in a computer system having a display device and **user** interaction device, involves displaying a number of query **object** class indicators, creating a number of query **objects** by repeatedly

selecting a query **object** class and entering query **object** data for each of the query **objects**, and **linking** the number of query **objects** by using the **user** interaction device to repeatedly **select** a first query **object**, a second query **object** and then **specify link** parameters, thus creating a **link object**. The query **objects** and the **link objects** are stored in a second database...

...The method further involves accepting a query specified in **text**, parsing the **text** query into a number of query **objects** and **link objects**, storing the query **objects** and **link objects** in the second database and displaying the query **objects** and the **link objects** on the display device...

....ADVANTAGE - Provides **user** with graphical, query software to directly manipulate database...

...Title Terms: **SELECT**;

International Patent Class (Main): **G06F-015/40**...

...**G06F-015/403**

Manual Codes (EPI/S-X): **T01-J05B4**...

...**T01-J12B**

46/3,K/100 (Item 100 from file: 350) **Links**
Derwent WPIX

THE APPLICANT

012228463 **Image available**
WPI Acc No: 1999-034570/199903
XRPX Acc No: N99-025913
Computerized method for visually depicting join relationships in database management system - involves accepting user command which enables to choose among join relationships visually depicted on display unit
Patent Assignee: INT BUSINESS MACHINES CORP (IBMC)
Inventor: LEVINE J M; LI K; MOCEK D J
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 5842209	A	19981124	US 96707140	A	19960903	199903	B

Priority Applications (No Type Date): US 96707140 A 19960903
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5842209	A		22	G06F-017/30	

Computerized method for visually depicting join relationships in database management system...

...involves accepting user command which enables to choose among join relationships visually depicted on display unit

...Abstract (Basic): The method involves visually depicting two tables, **fields** in the tables, and one or more **join relationships** between **fields** in the tables on a display unit coupled to a computer. The **join relationships** illustrate **join** operations performed between two or more tables in the database management system...

...A **user** command is accepted from a **user** input unit into the computer. The **user** command enables to **selectively choose** among the **join relationships** visually depicted on the display unit...

...ADVANTAGE - Provides full **SQL** functionality by allowing **user** to construct search **strings** with **logical relationships** between search parameters and create complex **SQL** commands by **linking** simpler **SQL** commands with **logical** conditional **operators**. Is implemented by one or more application programs interacting with **user** via **graphic user interface** under the control of the

operating system...
...Title Terms: **JOIN;**
International Patent Class (Main): **G06F-017/30**
Manual Codes (EPI/S-X): **T01-J05B4M...**

...**T01-J05B4P...**

...**T01-J12**

46/3,K/79 (Item 79 from file: 350) **Links**
Derwent WPIX

JAE APPLICANT

013958122 **Image available**
WPI Acc No: 2001-442336/200148
XRPX Acc No: N01-327204

Database management system for cross-node sharing of instructions, especially cached dynamic SQL statements in a multiple relational database management system environment

Patent Assignee: IBM CANADA LTD (IBMC); BIRD P M (BIRD-I); SNOWBELL M J (SNOW-I); INT BUSINESS MACHINES CORP (IBMC)
Inventor: BIRD P M; SNOWBELL M J
Number of Countries: 002 Number of Patents: 004
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
CA 2282999	A1	20010322	CA 2282999	A	19990922	200148	B
US 20020123978	A1	20020905	US 99404438	A	19990922	200269	N
CA 2282999	C	20030527	CA 2282999	A	19990922	200336	
US 6598058	B2	20030722	US 99404438	A	19990922	200354	N

Priority Applications (No Type Date): CA 2282999 A 19990922; US 99404438 A 19990922
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
CA 2282999	A1	E	62	G06F-017/30	
US 20020123978	A1			G06F-017/30	
CA 2282999	C	E		G06F-017/30	
US 6598058	B2			G06F-017/30	

Database management system for cross-node sharing of instructions, especially cached dynamic SQL statements in a multiple relational database management system environment

Abstract (Basic):
... and tables residing in a data partition. One of the nodes has a catalogue containing **metadata** describing tables and other **objects** and **relationships** between them in the data processing system. A global instruction cache at each node has a **statement** portion storing instruction **statements** and associated **executables**, each of which is designed for a particular environment. Instruction cache has dependency portion, which lists **objects** and **links** them to **corresponding executable** entries that depend on the **objects**. Shipped variation portion of the instruction cache has entries each referring to one **executable** entry and identifies the node from **statement** associated with **executable** was originally shipped.

...For **relational** database management systems, particularly for efficiently sharing instructions, especially cached dynamic **SQL statements** (and their sections) between applications connected at different nodes in a multiple node database while...

...Since identification of dynamic **SQL** entry in the shipping process is based on original node information, which **corresponds** to identification of entry at node the request was originally compiled and shipped from, compiled...

...Title Terms: **SQL**;

46/3,K/23 (Item 23 from file: 350) **Links**
Derwent WPIX

APPLICANT

0I6559228 **Image available**
WPI Acc No: 2004-717968/200470
XRPX Acc No: N04-569119

Data access providing method involves providing structured query language specification comprising logical fields for specifying criteria for data selection and specification of fields to be returned

Patent Assignee: INT BUSINESS MACHINES CORP (IBMC)
Inventor: DETTINGER R D; LAROCCA J L; STEVENS R J
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 20040193568	A1	20040930	US 2003401293	A	20030327	200470	B

Priority Applications (No Type Date): US 2003401293 A 20030327
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20040193568	A1		17	G06F-007/00	

Data access providing method involves providing structured query language specification comprising logical fields for specifying criteria for data selection and specification of fields to be returned

Abstract (Basic):
... A structured **query language** (**SQL**) specification comprising **logical fields** for specifying criteria for data selection and specification of **fields** to be returned, is provided for one of the applications. The mapping rules comprising access method, map **logical fields** to particular entity in **physical** data representation.
... For providing access to data having particular **physical** data representation...

...specification and the underlying data representation. The application defines data query requirement in a mode **abstract** manner, that are bound particular **physical** data representation at runtime...
...Title Terms: **FIELD**;

46/3,K/8 (Item 8 from file: 350) **Links**
Derwent WPIX

APPLICANT

017596935 **Image available**
WPI Acc No: 2006-108190/200611
Related WPI Acc No: 2003-778362; 2003-778410; 2005-294869
XRPX Acc No: N06-093925

Data access method for providing access to data having particular schema, involves accessing data abstraction model having logical field name that is different from physical field name of corresponding physical data as defined by schema

Patent Assignee: INT BUSINESS MACHINES CORP (IBMC)
Inventor: DETTINGER R D; JOHNSON P J; STEVENS R J; TONG I; WILL E
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 20060010127	A1	20060112	US 200283075	A	20020226	200611	B
			US 2005226181	A	20050914		

Priority Applications (No Type Date): US 200283075 A 20020226; US 2005226181 A 20050914
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20060010127	A1		14	G06F-017/30	Cont of application US 200283075

Data access method for providing access to data having particular schema, involves accessing data abstraction model having logical field name that is different from physical field name of corresponding physical data as defined by schema

Abstract (Basic):
... A data **abstraction model** with **logical field** definitions for **modeling** the data having particular schema e.g. **SQL** query, **XML query**, and exposing the data to users, is accessed in response to an **abstract** query. Each of the definitions comprises a **logical field** name that is different from the **physical field** name of the **corresponding physical** data as defined by the schema.
... Enables to access data independent of particular manner in which the data is **physically** represented...
...Title Terms: **ABSTRACT;**